BAKER BOTTS L.L.P.

30 ROCKEFELLER PLAZA

NEW YORK, NEW YORK 10112

TO ALL WHOM IT MAY CONCERN:

Be it known that we, Rafael Yuste, Vikram S. Kumar, Robert C. Froemke, and Paul Czkwianianc, citizens of Spain, the United States, the United States, and Poland, residing in the States of New York, Massachusetts, California and New York, whose post office addresses are: 601 West 113th Street Apt. 5G New York, NY 10025; 390 Commonwealth Avenue, Apt. 605, Boston, MA 02215; 1977 Pleasant Valley #6, Oakland, CA 94611; and 126 East 56th St., 19 Floor, New York, NY 10022, respectively, have invented an improvement in:

METHOD AND SYSTEM FOR ANALYZING FINANCIAL MARKET DATA

of which the following is a

SPECIFICATION

RELATED APPLICATION

[0001] This application claims priority from U.S. provisional application No. 60/245,132 filed on November 2, 2000, which is incorporated by reference herein in its entirety.

BACKGROUND OF INVENTION

[0002] The present invention relates to analyzing and interpreting datasets of financial market information. Examples of such datasets include closing price information for multiple financial instruments over time. As used herein, financial instrument means any commodity, security, instrument or contract traded on an open or closed market or exchange including stocks, bonds, options, future contracts, promissory notes and currencies.

[0003] It is often desirable to understand the relationship of various events occurring within a financial market information dataset. For example, share prices for various stocks may rise or fall with certain cohesiveness. It is desirable to determine which, if any, group of stocks ever exhibited correlated behavior (i.e. share prices rise or fall at the same time at least once in the period of observation), regularly exhibited correlated behavior (i.e. share prices rise or fall together on multiple occasions over the period of observation), and which stock, if any, consistently rises or falls before or after another stock rises or falls. It would also be advantageous to know the statistical significance of the relationships between the various events. In other words, whether the correlation among the various events is stronger than would be expected from random activity.

SUMMARY OF THE INVENTION

[0004] These and other advantages are achieved by the present invention which in one respect provides a method for analyzing a financial market dataset and for detecting relationships between various events reflected in the dataset.

[0005] In an exemplary embodiment, a method is presented for analyzing a financial market data array with a first dimension and a second dimension. The array is examined to detect events of interest, and those events of interest are stored in an event array having the same dimensions as the financial market data array, but the data in each element of the event array is binary. The financial market data array or the event array is then analyzed to determine relationships between the events of interest and correspondingly, relationships between the financial instruments corresponding to the financial market data.

[0006] In an additional exemplary embodiment, analyzing includes plotting a portion or all of the data in the first simplified array to allow visual examination of the relationships between the activities of interest. In another exemplary embodiment, the analysis step involves detecting events of interest that are coactive and determining whether the number of coactive events is statistically significant. This embodiment may include detecting all such coactive events (i.e. instances where events where events occur in at least two financial instruments simultaneously), detecting instances where many financial instruments are coactive simultaneously, or detecting instances where two or more financial instruments are each active in a certain temporal relationship with respect to one another (also referred to as coactivity).

[0007] In a further exemplary embodiment, the data analysis involves calculating a correlation coefficient between two financial instruments based on how often the financial instruments are coactive relative to how often the first financial instrument is active. Representations of all such financial instruments are displayed with lines between representations of the financial instrument having a thickness proportional to the correlation coefficient between the two financial instruments.

[0008] Another exemplary embodiment includes plotting a cross-correlogram or histogram of events of interest in a particular financial instrument with respect to events of interest in another financial instrument, so that the histogram will reveal the number of times an event of interest in the first financial instrument occurs a certain number of locations away from an event of interest in the second financial instrument. The cross-correlogram can be plotted with respect to only one financial instrument, thus showing

how many times an event of interest occurs before or after the occurrence of another event of interest in the same financial instrument.

[0009] Yet another exemplary embodiment includes displaying a time series "movie" showing activity occurring in one or more financial instrument relative to activity in a selected financial instrument. This "movie" is referred to herein as a spike triggered average. In this embodiment, a number of frames before and after events occurring in the selected financial instrument is chosen. A movie having the number of frames chosen is then displayed, with icons displayed for each non-selected financial instrument that was active within the chosen number of frames before or after activity occurring in the selected financial instrument. A parameter of the icon for each non-selected financial instrument, such as the color of the icon, is varied in each frame of the movie to correspond to the frequency that non-selected financial instrument is active and the corresponding number of frames before or after events occurring in the selected financial instrument.

[0010] Other exemplary embodiments include performing Hidden Markov Modeling on the event array to determine a hidden Markov state sequence and displaying a cross-correlogram between events of interest occurring in one region of interest while that region is in one of the detected Markov states and performing a singular value decomposition on the financial market data array.

[0011] In another aspect of the present invention there is provided a system for carrying out the foregoing method.

BRIEF DISCRIPTION OF THE DRAWINGS

[0012] For a more complete understanding of the present invention, reference is made to the following detailed description of exemplary embodiments with reference to the accompanying drawings in which:

Fig. 1 illustrates a flow diagram of a method in accordance with the present invention;

Fig. 2 illustrates a visual plot generated in accordance with the method of Fig. 1;

Fig. 3 illustrates an example of a data structure useful in the method of Fig. 1;

Fig. 4 illustrates a flow diagram of a method of analyzing data useful in the method of Fig. 1;

Fig. 5 illustrates a visual plot generated in accordance with the method of Fig. 1;

Fig. 6 illustrates a cross-correlogram generated in accordance with the method of Fig. 1;

Fig. 7 illustrates a correlation map generated in accordance with the method of Fig. 1;

Fig. 8 illustrates an exemplary format for displaying analysis results useful with the method of Fig. 1;

Fig. 9 illustrates another exemplary format for displaying analysis results useful with the method of Fig. 1;

Fig. 10 illustrates yet another exemplary format for displaying analysis results useful in the present invention; and

Fig. 11 illustrates yet another exemplary format for displaying analysis results useful in the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

[0013] Referring to Fig. 1, there is shown a flow diagram representing an exemplary method for analyzing data pertaining to financial instruments in accordance with the present invention. For purposes of this description, the financial instrument data is arranged in an input array corresponding to a time series of daily closing prices for various publicly traded stocks. Thus, the data array is a two dimensional array, with one dimension (indexed by a first dimensional index) corresponding to the different stocks and the other dimension (indexed by a second dimensional index) corresponding to the dates the closing prices were observed. The format of this input data array will be discussed further herein with reference to Fig. 3. It will be understood that the present invention is not limited to the particular data described. For example, the input data could correspond to any parameter of any type of financial instrument sampled at any frequency. For example, rather than including closing price data, the input data array could consist of price/earning ratios, market capitalization or trading volume of the various stocks over time. Alternatively, the data could consist of closing quoted prices for a commodity, such a electricity, available for delivery at a certain geographic location. Moreover, rather than consisting of daily closing prices, the data could consist of prices observed at the expiration of any other temporal period, such as every five minutes, or every month. Numerous other potential input data sets will be apparent to one of ordinary skill in the art.

[0014] In the exemplary embodiment, performance of the method is assisted by a general purpose computer with a processor adapted to operate the MAC-OS operating system and to interpret program code written in Interactive Data Language ("IDL")

version 5.1 or later, developed by Research Systems, Inc. The IDL program code of the exemplary embodiment is appended hereto as Appendices A, B and C described further herein. Other operating systems and programming languages could be used to perform the steps of the exemplary embodiment without departing from the scope of the invention, and the modifications necessary to make such a change will be apparent to one of ordinary skill in the art.

[0015] In step 101, events of interest in the input financial data array are detected. To further understand this step in the exemplary embodiment, reference is made to Fig. 3 where an example of an input data array 300 is shown. Data array 300 is a two dimensional array input data having multiple rows 322, 324 ... 326 and multiple columns 321, 323 ... 325. Each one of the rows 322, 324 ... 326 corresponds to a particular financial instrument, such as a particular stock. Thus, all data within a single row consists of observations corresponding to the same stock. Although only three rows are shown in Fig. 3, it will be understood that any number of rows could be present, the number of rows corresponding to the number of stocks under analysis. Each one of the columns 321, 323 ... 325 corresponds to a particular time period, such as a particular day on which the observation was made. Thus, all data within a single column consists of observations occurring during the same day. Although only three columns are shown in Fig. 3, it will be understood that any number of columns could be present, the number of columns corresponding to the number of observations made. Each data element, 301, 303, 305, 307, 309, 311, 313, 315, 317 corresponds to a particular observation. For example, data element 309 corresponds to the observation of the stock corresponding to

row 324 made during the period corresponding to column 323. Thus, data element 309 may contain the closing price of stock A observed on day X . In that scenario, data element 307 (which is in the same row as element 309) would contain the closing price of stock A observed during the period corresponding to column 321 and data element 315 (which is in the same column as element 309) would contain the closing price of the stock corresponding to row 326 observed on day X.

[0016] To assist in comparing the observations of different financial instruments trading at different prices, the data in input matrix 300 may be modified to contain percent change observations rather than actual closing price observations. For example, the closing price information for the stock associated with each row 322, 324 ... 326 of input data could be modified to contain percent change rather than absolute closing prices as follows. Beginning with the data element in the second column 323, the difference in closing price from the observation in first column 321 to the observation in second column 323 is calculated. The resulting difference is then divided by the closing price observation in the first column 321. The resulting value is stored in the data element in the second column 323. The process is repeated until the final column 325 is reached. Each element in the first column of data (i.e. data elements 301, 307 ... 313) is then set to zero. In this fashion, each data element will represent the percent change in closing price from the previous observation, rather than containing raw closing price data.

[0017] Returning now to Fig. 1, in step 101 the events of interest in the input data array 300 are detected. In one exemplary embodiment an event of interest is detected by calculating a statistical mean and standard deviation for all data elements corresponding

to a particular stock. Thus, where the input data is contained in the array 300, a mean and standard deviation is calculated for all data in each row of the simplified array. An event is then detected where the data element value exceeds the mean for all data in the row by a predetermined number of standard deviations. If activity were defined by a drop in value rather than an increase in value, the event could be detected by examining the data values in a financial instrument for an entry where the data element value is less than the mean for all data in the row by a predetermined number of standard deviations. The number of standard deviations may be entered by a user before the calculations are preformed, or a default number may be used, such as two or three. In this fashion, the method will detect those instances in time where the closing price is much higher than the average closing price, thus suggesting an event of interest has occurred.

[0018] In another exemplary embodiment, an event is detected by looking for a data value that exceeds a previous data values corresponding to the same stock instrument by a threshold amount. Thus, for example, if the closing price stored in data element 309 exceeded the closing price stored in data element 307 by a certain percentage, an event is said to have occurred at the time corresponding to data element 307. Again, if an event were indicated by a drop in value rather than an increase, the detection step would involve looking for a stock price that is less than previous stock price of the same stock by the threshold amount. The threshold amount can be specified by a user before the calculations are performed, or a default number can be used, such as five percent. The detection can occur over many time periods, for example, the closing price of a particular stock on day six could be compared to the stock's closing price on day one to see if an

increase beyond the threshold amount has occurred over that period. This would be useful to detect events that occur gradually over time rather than relatively instantaneously.

[0019] In step 103, the results of detection step 101 are stored in an event array. For this purpose, the event array is identical to the input array illustrated in Fig. 3; however, the data stored in the event array is binary rather than closing price values or percent changes. Thus, the entries in the event array would be 1 or 0 (or yes or no), corresponding to whether an event of interest occurred in the corresponding stock at the corresponding time.

[0020] In step 105, the stored data is analyzed. In one exemplary embodiment, the data is analyzed to determine whether various stocks are correlated (i.e. whether they are coactive), the strength of those correlations (i.e. how often they are coactive relative to how many times each stock or one of the stocks is active), how significant the correlations are (i.e. whether the correlation is stronger than would be expected if from a random data set) and the behavior of the entire observed stock population.

[0021] In the exemplary embodiment, the data is analyzed by plotting at least a portion of the data contained in the input data array 300. For example, stock price for one stock can be plotted over time. Stock prices for all observed stocks could also be plotted over time, either in separate plot windows or superimposed on the same plot window in either two or three dimensions. Additionally, the closing prices for all stocks could be averaged and plotted over time to show global behavior of the observed stocks. Fig. 2 illustrates

one possible plot of stock closing price over time, expressed as percent change as previously described.

[0022] In another exemplary embodiment illustrated in Fig. 5, the data is analyzed by plotting at least a portion of the data contained in the event array. As shown, a plot of events over time may be presented for one or multiple stocks in the input data set. For example, events occurring in three stocks are shown plotted versus time in Fig. 5. Events for each stock are plotted on separate horizontal axes 501, 503 ... 505. The vertical lines 507, 509, 511 represent events occurring at respective times in the corresponding stock.

[0023] In yet another exemplary embodiment illustrated in Fig. 4, the data in the financial data array is analyzed to determine the number of coactive events in the dataset and the statistical significance of those events. In step 401, a random distribution of stock price activity is generated. The random data is generated by shifting the data in each row of the input data array by a random amount. In step 403, the number of coactive events in the random dataset is counted. This process is repeated numerous times to generate a random distribution. The number of random trials may be set by the user or a default number of random trials may be conducted, such as 1000.

[0024] Counting coactive events for this purpose means counting all instances where two stocks are coactive. Coactive events for this purpose means events of interest that occurred in two stocks at the same time, or within a specified number of time intervals from each other. Thus, if the specified number of time intervals is one, then if a event occurred in the stock corresponding to row 322 at the time corresponding to column 321

(i.e. data element 301) and an event occurred in the stock corresponding to row 324 at the time corresponding to column 323 (i.e. data element 309), those events would be considered coactive. The time interval may be specified by a user before coactive events are counted, or may be a default setting such as two time intervals.

[0025] Once the random trials have been completed and a random distribution of coactive events generated, the actual number of coactive events in the data is calculated in step 405 using the same counting methodology was used to count coactive events in the random trials. The actual number of coactive events is then superimposed on a plot of the random distribution. The statistical significance of the coactive events is determined in step 407 by calculating the area under the distribution curve to the right of the number of actual coactive events in the data. This result, termed the "p-value" represents the probability that the number of detected coactive events in the actual data is produced by a random activity.

[0026] In a further exemplary embodiment, a random distribution of activity is generated as previously described, except the only coactive events that are counted in steps 403 and 405 are those where a predetermined number of stocks are coactive. The predetermined amount of coactive stocks may be specified by a user or a predetermined default value such as four may be used. Additionally, it may be specified whether exactly that many coactive events must be present or at least that many coactive events must be present to be considered a coactive event for counting. Thus, the embodiment allows instances of multiple simultaneously active stocks (rather than simply two simultaneously active stocks) to be counted and the statistical significance of that number

to be reported. In this exemplary embodiment, the random distribution and actual number of coactive events are plotted. The statistical significance of the actual number of coactive events is calculated using the formula: C_{rand}/N_{rand} where C_{rand} is the number of random trials that resulted in more coactive matches than the actual data set and N_{rand} is the total number of random trials used to generate the random distribution, and is reported to a user. Additionally, a chart may be drawn showing all observed stocks with line segments connecting those stocks that were coactive, such as the chart described herein with reference to Fig. 7.

[0027] In a still further exemplary embodiment, a random distribution of stock activity is generated as previously described except the only coactive events that are counted in steps 403 and 405 are those where at least two stocks are active a predetermined number times throughout the dataset. The number of times the two or more stocks must be active can be specified by a user or a default number such as two may be used. In this exemplary embodiment, the random distribution and actual number of coactive events are plotted. The statistical significance of the actual number of coactive events is calculated using the formula: C_{rand}/N_{rand} where C_{rand} is the number of random trials that resulted in more coactive matches than the actual data set and N_{rand} is the total number of random trials used to generate the random distribution, and is reported to a user. Additionally, a chart may be displayed showing all observed stocks with line segments connecting those stocks that were coactive, such as the chart described herein with reference to Fig. 7.

[0028] In yet another exemplary embodiment, a correlation map is plotted. To plot the correlation map, a correlation coefficient array is first generated for all of the stocks. The

correlation coefficients are defined as C(A,B)=number of times stock A and B are coactive divided by the number of times stock A is active. For this purpose, coactive means active at the same time, or within a specified number of time intervals of each other. The number of time intervals may be specified by a user or a default number such as one time increment may be used. The number of correlation coefficients will be equal to the square of the number of stocks observed. A correlation map is then drawn consisting of a map of all stocks with lines between each pair of stocks having a line thickness proportional to the correlation coefficient of those two stocks. An example of such a correlation map is illustrated in Fig. 7. There, an icon representing each observed stock 701, 703, 705, 707, 709, 711 is plotted around a circle 713. The thickness of line 717 is proportional to the magnitude of the correlation coefficient for stocks 701 and 709. Line 715, which appears thicker than line 717, indicates that the correlation between stocks 705 and 709 is stronger than the correlation between stocks 701 and 709. Similarly, line 719, which appears thicker than lines 715 or 717, indicates that the correlation between stocks 701 and 705 is stronger than the correlation between stocks 701 and 709 or stocks 705 and 709. If the correlation coefficient is below a predetermined threshold amount, the corresponding line may be omitted from the correlation map. The predetermined threshold amount may be specified by a user or a default threshold may be used.

[0029] In still another exemplary embodiment, a cross correlogram is drawn to show potential causality among stock activity. This can be used to find stocks with events that consistently precede or follow events of another stock. A cross correlogram simply

creates a histogram of the time intervals between events in two specified stocks. A line of height proportional to the number of times the second stock is active one time interval following activity by the first stock is plotted at +1 on the x-axis of the histogram. A line of height proportional to the number of times the second stock is active two time intervals following activity by the first stock is plotted at +2 on the x-axis of the histogram, and so on. An example of such a cross correlogram is illustrated in Fig. 6. The line 601 represents the number of occasions the first and second stocks were active at the same time, while line 607 represents the number of times the second stock was active three time intervals after the first stock was active. A cross correlogram may be plotted for a single stock to detect temporal characteristics in the stock's activity such as the fact that the stock is active with a period of every three time intervals a certain number of times during the period of observation.

[0030] IDL code implementing all of the preceding steps of the exemplary embodiment is attached hereto as Appendix A. The procedure "MultiStock" and "MultiStock_event" are the main procedures. All relevant sub-procedures and functions are also included in Appendix A.

[0031] An exemplary embodiment related to the cross-correlogram provides for displaying what is referred to as a "spike triggered average", which consists of a time series "movie" showing activity occurring in one or more stocks under investigation relative to activity in a selected stock. In this embodiment, a particular reference stock is selected. A data window consisting of a number of frames before and after events occurring in the selected stock (known as primary events) is then chosen or a default

number of frames may be used, such as ten. In the event ten frames are chosen, the resulting movie will consist of twenty-one frames, ten frames corresponding to the ten time periods before each event occurring in the reference stock, one frame corresponding to the time of each event in the reference stock and ten frames corresponding to the ten time periods after each event in the reference stock.

[0032] Each frame of the movie will consist of a representation of all stocks under investigation. An example of such a frame is shown in Fig. 8. There, frame 800 consists of several icons 801, 803, 805, 807, 809 and 811, each corresponding to a stock under investigation. Each icon may be a solid square. The representations may also include ticker symbols 802, 804, 806, 808, 810 and 812 to further identify the stocks under investigation. A parameter of the icon for each stock, such as the color of the icon, is varied in each frame of the movie. The parameter varies in each frame to correspond to the frequency that events occur in the stock under investigation (known as secondary events) at the corresponding number of time periods before or after an event occurs in the reference stock.

[0033] For example, if the reference stock selected had respective events at times t=20 and t=50 and a movie length of twenty-one frames was selected, corresponding to ten frames before and ten frames after each primary event (i.e. an event in the reference stock), the movie would appear as follows. The first frame would be derived based on events occurring in the stocks under investigation at time t=10 and t=40 (i.e. 10 time periods before the respective events in the reference stock). Thus, if the first stock under investigation had an event at time t=10 and t=40, the icon parameter for that stock that is

displayed in the first frame would correspond to an event always occurring ten frames before an event in the reference stock, for example the icon color may be red. If the stock under investigation instead had an event at time t=10, but not at time t=40, the icon parameter for that stock that is displayed in the first frame would correspond to an event occurring half the time ten frames before an event in the reference stock, for example the icon color may be orange. The process is repeated for each stock under investigation for each of the frames in the spike triggered average movie. The resultant movie will illustrate the frequency that events occur in the stocks under investigation at the corresponding number of time periods before or after events occurring in the reference stock. This information may be used to uncover possible causality in the temporal domain among the stocks by identifying stocks whose activity appears to trigger or be triggered by activity in other stocks.

[0034] In a still further exemplary embodiment, the data is analyzed in step 105 of Fig. 1 by finding a hidden Markov state sequence from the event array. This embodiment uses the principal of Hidden Markov modeling described in Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings of the IEEE, vol. 77 pp. 257-286 (1989), which is incorporated by reference herein. Essentially, a Markov model is a way of modeling a series of observations as functions of a series of Markov states. Each Markov state has an associated probability function which determines the likelihood of moving from that state directly to any other state. Moreover, there is an associated initial probability matrix which determines the likelihood the system will begin in any particular Markov state. In a hidden Markov

Model, the Markov states are not directly observable. Instead, each state has an associated probability of producing a particular observable event. A complete Markov model requires the specification of the number of Markov states (N); the number of producible observations per state (M); the state transition probability matrix (A), where each element a_{ij} of A is the probability of moving directly from state i to state j; the observation probability distribution matrix (B), where each element $b_i(k)$ of B is the probability of producing observation k while in state i; and the initial state distribution (P), where each element p_i of P is the probability of beginning the Markov sequence in state i.

[0035] In the exemplary embodiment, it is assumed that the number of times events occur in a stock within each Markov state follows the Poisson distribution. Thus, each stock in each state has an associated Poisson Lambda parameter, which can be understood in the exemplary embodiment to correspond to the rate at which events occur in the stock. The set of all of these Lambda parameters is then assumed to be the B matrix. Given the estimations of the Markov Model parameters, the method uses the Viterbi algorithm to find the single best state sequence, i.e. the sequence of Markov states that most likely occurred to generate the observed results. The number of Markov states N may be selected by the user, or a default number such as six states may be used. The Viterbi algorithm is described as follows:

Initialization:

$$\delta_1(i) = p_i b_i(O_1) \quad 1 \le i \le N, \qquad (1)$$

$$\psi_1(i) = 0, \tag{2}$$

Recursion:

$$\delta_t(j) = \max_{1 \le i \le N}\left[\delta_{t-1}(i)a_{ij}\right] \; b_i(O_t) \quad 2 \le i \le T \tag{3}$$

$$1 \le j \le N,$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}\left[\delta_{t-1}(i)a_{ij}\right] \quad 2 \le t \le T \tag{4}$$

$$1 \le j \le N,$$

Termination:

$$p^* = \max_{1 \le i \le N}\left[\delta_T(i)\right], \tag{5}$$

$$q_T^* = \arg\max_{1 \le i \le N}\left[\delta_T(i)\right], \tag{6}$$

Path (backtracking):

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right) \quad t = T-1, T-2, \cdots, 1. \tag{7}$$

[0036] In the algorithm, $\delta_t(i)$ represents the highest probability along a single path through all possible Markov state sequences up to time t that accounts for the first t observations (O_t) and ends in state i. ψ is used to store the argument which maximizes $\delta_t(i)$. Once a possible state sequence q_t^* is generated, the state sequence plot can be generated such as the one shown in Fig. 9. In that example, six states are shown, corresponding to horizontal lines 901, 903, 905, 907, 909, 911. Each point on the plot

represents the Markov state the model is in at the relevant time. For example, point 913 represents the Markov model being in state 903 while point 915 represents the model being in state 907. Each different state represents differing behavior of the stocks. For example, one group of stocks may exhibit events of interest more frequently than the remaining stocks when the model is in the first state 901, while those same stocks may exhibit fewer or no events when the model is in the second state 903. Correspondingly, another group of stocks may exhibit more frequent events of interest while in the third state 905 than other stocks and fewer events of interest while in the fourth state 907.

[0037] A cross-correlogram between stocks in a selected state can be plotted using the methodology previously described, where only event data corresponding to the time the model is in the selected state is used in generating the cross-correlogram. The state may be selected by the user or a default state such as the first state may be used.

[0038] IDL code implementing the preceding embodiment involving the hidden Markov model is attached hereto as Appendix B. The procedure "hidden_markov" and "hidden_markov_event" are the main procedures. All relevant sub-procedures and functions are also included in Appendix B.

[0039] In a yet further exemplary embodiment the data is analyzed by performing a singular valued decomposition (SVD) on the data in the input stock data array, such as that shown in Fig. 3. In this embodiment, it is not necessary to detect events or store events in an event array. A singular valued decomposition takes advantage of the fact that in some sets of data produced from N different sources, such as N different stocks,

some of the stocks will not be creating independent data. In other words, there may be degeneracy in the data, which allows the data set to be decomposed into a number of eigenmodes i.e., orthogonal eigenvectors, with the eigenvalue (or singular value) representing the weight of the eigenvector in the system.

[0040] In a singular valued decomposition, the data set is reduced from N dimensions, where N is the number of selected stocks, to *d* dimensions, where *d* is the specified number of eigenmodes and is less than N. The SVD algorithm, which is well known to one of ordinary skill in the art and is specified in the code in Appendix C, fits the observed stock data to a data model that is a linear combination of *d* number of functions of the spaces of data (such as time and stock price). Since *d* is specified rather than calculated by looking for degeneracy in the data, the resultant decomposition constitutes an approximation. Minimizing the sum of the squares of the errors in the approximation to the model, the SVD algorithm discards the eigenmodes corresponding to the smallest N-*d* eigenvalues.

[0041] The stock data may be preprocessed before the SVD is performed by subtracting the median from each stock's closing price data. In other words, for each stock, a median is calculated and subtracted from each closing price entry for that stock. Additionally, when a positivity constraint is employed in the SVD algorithm (i.e. when only stock prices rising above the baseline are considered) an absolute value of the resultant data may be taken to ensure that downward events (i.e. drops in stock prices below the baseline) are considered in performing the SVD.

[0042] In this embodiment, the result that is plotted for visual analysis may be the level of each stock's contribution to each of the calculated *d* eigenmodes. For example, the result may be displayed in the format shown in Fig. 8, with each stock represented by an icon 801, 803, 805, 807, 809 and 811 and optionally a ticker symbol 802, 804, 806, 808, 810 and 812. A parameter of the icon, such as its color, may be adjusted to represent the level of the stock's contribution to the displayed eigenmode. A separate plot can be generated for each of the calculated *d* eigenmodes.

[0043] Alternatively, a plot, such as that shown in Fig. 10 may be generated to display the results of the SVD. This plot 1000, which displays singular values on the y-axis and mode number on the x-axis, represents the power of each mode in explaining the variance of the data set (i.e. the strength with which each of the calculated modes explains the tendency of the stock prices to deviate from the baseline). The example plot 1000 shows that most of the variance is explained by mode 0 (1006), mode 1 (1007) and mode 2 (1008), while modes 3 (1009), 4 (1010) and 5 (1011) explain little of the activity in the data set.

[0044] A third visualization useful to show the result of the SVD is shown in Fig. 11. In that example, three windows 1101, 1003 and 1005 are shown. The user first selects the mode for which data should be displayed, such as by using the slider bar 1119. In the top window 1101, an icon for each stock (e.g. 1107, 1009) in the data set is displayed, with the stock's position on the y-axis corresponding to the strength with which that stock participates in the selected mode. The middle window 1103 shows a time series representation of the selected mode. In other words, window 1103 displays the aggregate

stock activity corresponding to the selected mode. The bottom window 1105 is a superimposed plot of all of the stocks participating in the selected mode. As can be seen, the spike occurring around time day 300 (1115) in the bottom plot 1105 corresponds to the spike occurring at the same time (1111) in the aggregate mode activity shown in the middle plot 1103. Similarly, the spike occurring around day 480 (1117) in the bottom plot 1105 corresponds to the spike occurring at the same time (1113) in the middle plot 1103. Thus, it can be seen that activity in the identified stocks shown in the bottom plot 1105 does constitute the activity of the mode shown in the middle plot 1103.

[0045] IDL code implementing the preceding embodiment involving the singular value decomposition algorithm is attached hereto as Appendix C. The procedure "ssvd_gui" and "ssvd_gui_event" are the main procedures. All relevant sub-procedures and functions are also included in Appendix C.

[0046] Although the present invention has been described in detail with reference to exemplary embodiments thereof, it should be understood that various changes, substitutions and alterations can be made hereto without departing from the scope or spirit of the invention as defined by the appended claims.

APPENDIX A

```
pro choose_correl
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common things_com, state3

      base4=WIDGET_BASE(/ROW, title='Choose pvalue & windowsize')
      field1=CW_FIELD(base4, /RETURN_EVENTS, /FLOATING, TITLE='p value for chi
squared',$
            VALUE=.05, UVALUE=0)
      field2=CW_FIELD(base4, /RETURN_EVENTS, /INTEGER, TITLE='window size, ONLY
ODD',$
            VALUE=1, UVALUE=0)
      button1=WIDGET_BUTTON(base4, VALUE='Go Cross Correlate', UVALUE=2)
      WIDGET_CONTROL, /realize, base4
      state3={field1:field1, field2:field2, button:button1}
      WIDGET_CONTROL, WIDGET_INFO(base4, /CHILD), SET_UVALUE=state3

      xmanager, 'choose_correl', base4
 end
```

```
; NAME:
;     choose_correl_event, event
; SYNOPSIS:
;     choose_correl_event, event
; DESCRIPTION:
;     This handles the events for choose_correl.  Together they allow the user
to
; find the cross correlation coefficients and specify a p value and bin size.
; As a result, the cross correlation matrix is printed to a file in the working
directory
;_____________EVENT HANDLER_____________________________________
 pro choose_correl_event, event

      ;__________________global variables_____________________________

      common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
      common things_com, state3
      common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined

   stateholder3=WIDGET_INFO(event.handler, /CHILD)
   WIDGET_CONTROL, stateholder3, GET_UVALUE=state3
   WIDGET_CONTROL, state3.field1, GET_VALUE=set_p_value
   WIDGET_CONTROL, state3.field2, GET_VALUE=winsize

 ;------calculate the correlation coefficients for all the cell pairs------

      if ((set_p_value lt 0) OR (set_p_value gt 1)) then begin
            mess=WIDGET_MESSAGE('The p must be between 0 and 1!', /ERROR)
      endif else begin
      if (winsize ge frame_no) then begin
```

```
            mess=WIDGET_MESSAGE('The window size cannot be larger than the total
number of frames in the movie!', /ERROR)
      endif else begin
      if ((winsize mod 2) eq 0) then begin
            mess=WIDGET_MESSAGE('Please enter an odd window size for
uniformity', /INFORMATION)
      endif else begin
      coef=correl_coeff(yes_no_values, p=set_p_value, cells=cell_no,$
             win=winsize, frame_no)
      close, 1
      openr, 1, 'correl.dat'
      stat_stuff=fstat(1)
      file_size=stat_stuff.size
      close, 1
      print, file_size
      ;if (file_size gt 8112) then begin
      ;     mess=WIDGET_MESSAGE('File is too large to display through a widget.
Open it manually.  If you did not save it, it is named correl.dat',
/INFORMATION)
      ;     correl_coef_defined=1
      ;endif else  begin
      xdisplayfile, 'correl.dat', title = "Correlation Coefficient Matrix",
group = event.top, width = 85, height = 45
      correl_coef_defined=1
      file2=pickfile(/write, file='Correlation_Coeff_exp_#_')
      if (file2 eq '') then begin
            mess=WIDGET_MESSAGE('The Correlation Coefficient Matrix has not been
saved.', /INFORMATION)
      endif else begin
            get_lun, lun1
            openw, lun1, file2
            printf,lun1, coef
            free_lun, lun1
            close, lun1
      ;endelse
      endelse
      endelse
      endelse
      endelse
end
```

```
pro choose_crosscorr
base=Widget_base(/column)

button1=widget_button(base, value='Standard Cross Correlogram',uvalue=1)
button2=widget_button(base, value='Cell firing rate  Cross
Correlogram',uvalue=2)
widget_control,base, /realize

xmanager, 'choose_crosscorr', base
end
```

```
pro choose_crosscorr_event, ev
widget_control, ev.id, get_uvalue=uval
case uval of
      1 : draw_cross
      2 : draw_cross2
endcase
```

```
end
; NAME:
;     choose_threshold
; DESCRIPTION:
;     This program creates the widget with which the user can change the
thresholds for the
; detection of spikes in cells.  It is called by calling Find Spikes to
Multicell.
; It uses MAKE_BINARY (see choose_threshold_event) to make the binary array.
pro choose_threshold

      ;_____________THE GLOBAL VARIABLES_____________

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common things_com, state3

  base5=WIDGET_BASE(/Column, title='Choose threshold for spike detection')
  field1=CW_FIELD(base5, /RETURN_EVENTS, /FLOATING, TITLE='Threshold for 2
consec. frames to be a spike',$
      VALUE=thresh1, UVALUE=0)
  field2=CW_FIELD(base5, /RETURN_EVENTS, /FLOATING, TITLE='Threshold for 3
consec. frames to be a spike',$
      VALUE=thresh2, UVALUE=0)
  button1=WIDGET_BUTTON(base5, VALUE='Find Spikes', UVALUE=2)
  WIDGET_CONTROL, /realize, base5
  state3={field1:field1, field2:field2, button:button1}
  WIDGET_CONTROL, WIDGET_INFO(base5, /CHILD), SET_UVALUE=state3
  xmanager, 'choose_threshold', base5

 END
```

```
; NAME:
;     choose_threshold_event
; DESCRIPTION:
;     This is the event handler for the choose_threshold widget.
; With this, the user can change the threshold value used in determining which
increases in
; Calcium intensity correspond to action potentials.It uses make_binary.pro to
convert the
; pixel or deltaF arrays into 0 or 1s.
;_____________EVENT HANDLER_____________________________________

pro choose_threshold_event, event
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common things_com, state3
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined

      stateholder3=WIDGET_INFO(event.handler, /CHILD)
      WIDGET_CONTROL, stateholder3, GET_UVALUE=state3
```

```
      WIDGET_CONTROL, state3.field1, GET_VALUE=thresh1
      WIDGET_CONTROL, state3.field2, GET_VALUE=thresh2

      if ((thresh1 eq 0) OR (thresh2 eq 0)) then begin
            mess=WIDGET_MESSAGE('Threshold cannot be 0!', /Error)
      endif else begin
            WIDGET_CONTROL, event.top, /hourglass
            for cell = 0, cell_no - 1 do begin
                  threshold1[cell] = thresh1
                  threshold2[cell] = thresh2
            endfor
            yes_no_values=make_binary(pixel_array, frame_no, cell_no,
threshold1, threshold2)
            spikes_defined=1                              ;this lets other programs
now that spikes have been defined
            mess=WIDGET_MESSAGE('Done finding spikes boss.', /INFORMATION)
      endelse
end
```

```
; NAME:
;     contingency_values
; DESCRIPTION:
;     This program creates a contingency table for the data of two spike trains
and from it
; gives a chi squared value.
;
; METHOD: observed_array is an array of 4 elements they are as follows:
;     observed_array(0)=number of hit_hits, i.e the number of times the first
cell and the second
; cell have spikes at exactly the same time.
;     observed_array(1)= number of observed hit_misses
;     observed_array(2)= number of observed miss_hits
;     observed_array(3)= number of observed miss_misses
;
; This is called by CORREL_COEFF.PRO to calculate the correlation coefficients
between all  the cells of
; an analysis.  Using the contingency table, we get a chi squared value which we
compare with the chi squared
; value created when the user specifies the p for the data to be significant.
Each pair of cells has its
; own contingency value.

function contingency_values, filearray1, filearray2

;this program will print out a contingency table and then print out the chi-
square value.
;it also print out the probability that the null hypothesis is true

;Null hypothesis: The two files are independant
;Constraint: THIS WILL ONLY WORK IF THE SPIKE TRAINS FOR THE CELLS ARE IN TERMS
OF 0 AND 1
       ;declaring the arrays
      observed_array=fltarr(4)
      expected_array=fltarr(4)
      result=fltarr(4)

      ;This makes a spike for cell1 = 2 where a spike for cell2=1. Therefore
cell1-cell2 will be 0 only
```

```
      ;at times when both cell1 and cell2 have a 0 (at miss_miss). A
      ;Due to this, a hit_miss can be found by seing where the above difference
array is equal to 1.

      ;eg., 1 0 0 1 0 1 -> 2 0 0 2 0 2

      filearray1=2*filearray1

      ;the following array is used to calculate all of the values needed in the
table

      ;we subtract the spike trains of the two cells into difference_array.
      ;therefore if difference_array has a 0, there

      difference_array=filearray1-filearray2

      ;this is the m(1,1) values or the hit_hit
      ;a spike in filearray1 will be a 2. That in filearray2 will be a 1.
Therefore, in the difference array
      ;a hit_hit will be represented by a (2-1) or a 1
      arr_hit_hit=difference_array eq 1
      if (total(arr_hit_hit) gt 0) then begin
            observed_array(0)=total(arr_hit_hit)
      endif else begin
            observed_array(0)=0
      endelse


      ;this is the m(0,0) values or miss_miss
      arr_temp3=difference_array eq 0
            if (total(arr_temp3) eq 0) then begin
                  observed_array(3)=0
            endif else begin
                  miss_miss_data=total(arr_temp3)
                  observed_array(3)=miss_miss_data
            endelse

      ;observed_array(3)=n_elements(difference_array(where(difference_array eq
0)))

      ;this is the m(1,0) values or the hit_miss, i.e the number of times when
at an instant in time,
      ;the first cell has a spike, but the second cell does not.
      arr_hit_miss=difference_array eq 2
            if  (total(arr_hit_miss) eq 0 )then begin
                  observed_array(1)=0
            endif else begin
                  hit_miss_data=total(arr_hit_miss)
                  observed_array(1)=hit_miss_data
            endelse

      ;this is the m(0,1) values or the miss_hit, i.e the number of times when
at an instant in time, the
      ;second cell has a spike while the first cell does not. (cell1(t)=0, but
cell2(t)=1)
      arr_miss_hit=difference_array lt 0
            if (total(arr_miss_hit) eq 0) then begin
```

```
            observed_array(2)=0
        endif else begin
            miss_hit_data=total(arr_miss_hit)
            observed_array(2)=miss_hit_data
        endelse

    sum_first_row=observed_array(0)+observed_array(1)
    sum_second_row=observed_array(2)+observed_array(3)
    sum_first_col=observed_array(0)+observed_array(2)
    sum_second_col=observed_array(1)+observed_array(3)

    n_total=sum_first_row+sum_second_row
    ;now we calculate the observed and expected values for chi squared

    ;for j=1, i=1
     expected_array(0)=(sum_first_row*sum_first_col)/n_total
    ;for i=1, j=0
     expected_array(1)=(sum_first_row*sum_second_col)/n_total
      ;for i=0, j=1
     expected_array(2)=(sum_second_row*sum_first_col)/n_total
    ;for i=0, j=0
     expected_array(3)=(sum_second_row*sum_second_col)/n_total

     for k=0, 3 do begin
        result(k)=((observed_array(k)-expected_array(k))*(observed_array(k)-
expected_array(k)))/expected_array(k)
     endfor
     answer=total(result)
    return, answer
end
```

```
; NAME: correl_coeff.pro
; INTRODUCTION:
;     This routine is used to output a correlation coeff. matrix with the
insignificant
; correlation coeff =0. It uses the functions Contingency_values and
find_matches
;
; INPUTS:
;     p_value-> This is the probability that a random variable is greater than
the cut off value in
; a chi squared distribution. For example, p=.05 means the cut off value should
be high enough
; so there is only a 5 percent chance that any random number can be greater.
;
;     cell_number-> the number of cells chosen from the slice.
;
;     winsize-> the size of the window to look for coincident spike in. The
default is 1
;
;
; DESCRIPTION:
;     The function does the following:
;          1. Uses the IDL function chi_sqrcvf to find the cut off value for
the input p value. It is
; called as chi_sqrcvf(p_value, degrees of freedom)
```

```
;           2. Creates a contingency table for a pair of cells by the home-made
function
; contingency_values. pro. From this table, it then finds the chi squared value
for the null hypothesis that
; the cells are independant.
;           3. It prints out all of the correlation coefficients that are
significant (fail the null
; hypothesis, but pass the cut_off test) and indicate the cells of the pair are
dependant.
;
; OUPUT:
;     a matrix of the filtered correlation coefficients.
;NOTE: To see all of the cross correlation coefficients, make the p_value=1!

function correl_coeff, yes_no_values, p=p_value, cells=cell_number, win=winsize,
frame_no

      forward_function contingency_values, find_matches
      output_array=fltarr(cell_number, cell_number)


 ;----------------------------FIND THE CUT OFF VALUE---------------------------
-------------

      limit=chisqr_cvf(p_value, 1)           ;the degree of freedom is 1 in our
case

 ;----------------------------CONTINGENCY TEST AND CORREL COEFF CALC.-----------
-------------

bin_no=frame_no/winsize                    ;using the window size, this is the
number of bins
left_overs=frame_no mod winsize                         ;these are the frames that
didn't fit in the bins, but are still in the array (at the end)
total_elements=bin_no+left_overs                ;this is the total number of
elements in the binned array
if (total_elements lt 1) then begin
      mess=WIDGET_MESSAGE('Window size too large or two few frames.', /ERROR)
endif else begin
close, 1

IF (winsize eq 1) THEN BEGIN                            ;IF THE WINDOW SIZE IS
ONE, NO BINNING IS DONE!
      for i=0, cell_number-1 do begin
            for j=0, cell_number-1 do begin
                  temp_value=contingency_values(yes_no_values(i, *),
yes_no_values(j, *))
                  if (temp_value ge limit) then begin
                        output_array(i,
j)=find_matches(yes_no_values(i, *),yes_no_values(j, *),$
                         winsize)
                  endif
      endfor
endfor
ENDIF ELSE BEGIN
```

```
cell1_binned=intarr(total_elements)
cell2_binned=intarr(total_elements)
stop_index=-1
index=0                                                                    ;in case
no bins are made, then the stop index used in the second for loop below will be
used
for i=0, cell_number-1 do begin
      for j=0, cell_number-1 do begin
            for bin_number=0, bin_no-1 do begin
;making the binned cell array
                  start_index=bin_number*winsize
         ;this is the starting point for a bin
                  stop_index=start_index+winsize-1
;this is the ending point for a bin

      cell1_binned(bin_number)=total(yes_no_values(i,start_index:stop_index))
;we are binning the input arrays for the contingency table

      cell2_binned(bin_number)=total(yes_no_values(j,start_index:stop_index))
;for the second cell being looked at
            endfor
            index=0
            for bin_number=stop_index+1, frame_no-1 do begin
;add the elements that don't fit in the bins
                  cell1_binned(bin_no+index)=yes_no_values(i, bin_number)
                  cell2_binned(bin_no+index)=yes_no_values(j, bin_number)
                  index=index+1
            endfor
                  cell1_binned=cell1_binned ge 1
         ;this converts the arrays into binary ones (eg if three elements in a
bin each have a value 1 then the total will be 3 that will be converted to 1
here)
                  cell2_binned=cell2_binned ge 1
                  temp_value=contingency_values(cell1_binned, cell2_binned)
                  if (temp_value ge limit) then begin
                       output_array(i, j)=find_matches(yes_no_values(i,
*),yes_no_values(j, *),$
                                       winsize)
                  endif
      endfor
endfor
ENDELSE
close, 1
openw, 1, 'correl.dat'              ;CHOOSE_CORREL USES XDISPLAYFILE AND THIS
FILE TO WRITE
printf, 1, output_array
close, 1
return, output_array                ;this is the correl coeff matrix
endelse
end
```

```
; NAME:
;       CORREL_MAP_IMAGE_PLANE
;
; PURPOSE:
;       This procedure creates a circular representation of correlation between
cells of the slice.
```

```
;     Those cells that
;     are correlated in either direction are joined by lines which are
proportional to their
;     correlation coefficients

pro correl_map_image_plane, coef_array
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size

      for s=0, cell_no-1 do begin
            ;; 1) draw boxes over active cells
            ;; we only worry about the active cells- cells that have at least
one spike
            if total(yes_no_values(s,*)) gt 0 then begin
                  x=location(s).coord(0)  ;the x axis of where you clicked on
cell
                  y=location(s).coord(1)  ;the y axis of where you clicked on
cell
                  for j=-box_size, box_size do begin
                        plots, [x+j, x+j], [y-box_size, y+box_size], /device
                  endfor
                  xyouts, x-50, y, s+1, font=-1, charsize=1.9, /device
                  ;; 2) draw lines between 'correlated' cells
                  for t=0, cell_no-1 do begin
                        if (coef_array(s, t) gt 0) then begin
                              plots,
[location(s).coord(0),location(t).coord(0)], [location(s).coord(1),
location(t).coord(1)],$
                              thick=2*coef_array(s, t), /device
                        endif
                  endfor
            endif
      endfor
      !p.font=0
end
```

```
; Name:  count_random_hits_2_manyX, least_no_of_matches
; Synopsis: count_random_hits_2_manyX,random_array, least_no_of_matches
; Description: This program tests to see how many times more than
least_no_of_matches two random cells
;     fire simultaneously.
; Description of variables:
;     yes_no_values: the binary data
;     random_array:  a random array created by RANDOM_TEST looking at the spikes
in yes_no_values
;     least_no_of_matches: Minimum number of times that two cells must fire to
be considered a coactive pair
;     window_size: number of frames before and after to look for a coincident
spike in the other cell

function count_random_hits_2_manyX, random_array, least_no_of_matches,
window_size, num_active_cells

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
```

```
      total_frame_no, time_resolution, x_size, y_size, box_size

      random_hits=0

      for cell_1_counter = 0, num_active_cells - 2 do begin
            for cell_2_counter = cell_1_counter + 1, num_active_cells - 1 do
begin
                  temp_hits = 0
                  for window_counter = -window_size, window_size do begin
                        temp_cell_1 = intarr(frame_no + (2 * window_size))
                        temp_cell_2 = intarr(frame_no + (2 * window_size))
                        temp_cell_1(window_size:frame_no+window_size-1) =
random_array(cell_1_counter, *)
                        temp_cell_2(window_size:frame_no+window_size-1) =
random_array(cell_2_counter, *)
                        temp_cell = (temp_cell_1 * shift(temp_cell_2,
window_counter))
                        temp_hits = temp_hits + total(temp_cell)
                  endfor
                  if (temp_hits ge least_no_of_matches) then begin
                        random_hits = random_hits + 1
                  endif
            endfor
      endfor

      return, random_hits
end
```

```
; Name:  count_random_matches
; Description: this program takes in a random array of spike trains and counts
the number of sets of coactive spikes
;      (where the number of spikes in a set is 'times_repeated', e.g. if 2, we
count the number of all possible pairs
;      of coactive spikes
;
; Defintion of variables:
;     random_array is the array of the location of spikes for the simulated data
;     times_repeated is the number of times the spikes are clustered together in
calculating the
;           combination. For example, if looking at pairs of two,
times_repeated=2. If looking for cells firing 3X together, it is 3.
;
; Formula Used:  C(n,r)=  n!/(r! * (n-r)!)
;

function count_random_matches, random_array, times_repeated, window_size

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size

      ;find the number of hits for the simulated data.
      ; we can do this by summing the rows (cells) of one column (frame) since
spikes are
      ; represented by 1 and then finding the combinations of this taken
times_repeated at a time
      ; eg. cell 1 :00110011010101010100101010
      ;        cell 2: 00100010000100000010000001
```

```
      ; If in frame 23, only cells 2, 3, and 8 fire, the sum of the frames will
      ; be 3 and then the combination taken two at a time is C(3,2)=3

      random_matches = 0

      if (window_size eq 0) then begin
            for frames = 0, frame_no - 1 do begin
                  matches = total(random_array(*, frames))       ;finds n or the
number of hits at this frame
                  if ((matches gt 0) and (matches ge times_repeated)) then begin
                        ;; find number of possible pairs
                        matches_pair = factorial(matches) /
(factorial(times_repeated) * factorial(matches-times_repeated))
                  endif else begin
                        matches_pair = 0                    ;cant find factorial
of a negative number
                  endelse
                  random_matches = random_matches + matches_pair
            endfor
      endif else begin
            ;; first compress spike trains according to window size
            binned_random_array = intarr(n_elements(random_array[*,0]), frame_no
- (window_size * 2))
            for frame_counter = window_size, frame_no - (window_size + 1) do
begin
                  for window_counter = -window_size, window_size do begin
                        for cell_counter = 0, n_elements(random_array[*,0]) - 1
do begin
                              if (random_array[cell_counter, frame_counter +
window_counter] ge 1) then begin
                                    binned_random_array[cell_counter,
frame_counter - window_size] = 1
                              endif
                        endfor
                  endfor
            endfor
            ;; then look for matches in the binned spike trains (although a
single spike can fall into many bins)
            for frame_counter = 0, n_elements(binned_random_array[0, *]) - 1 do
begin
                  matches = total(binned_random_array(*, frame_counter))
;finds number of hits at this frame
                  if (matches ge times_repeated) then begin
                        matches_pair = factorial(matches) /
(factorial(times_repeated) * factorial(matches - times_repeated))    ; finds
number of possible pairs
                  endif else begin
                        matches_pair = 0                    ;cant find factorial
of a negative number
                  endelse
                  random_matches = random_matches + matches_pair      ;increases
the matches count for the random data by that of this frame
            endfor
      endelse

      return, random_matches
end
```

```
; Name:  count_random_matches_many
; Description: this program takes in the actual binary spike train and the
random one and sees how
; the actual one compares to the random one.  To test the significance, it looks
at the number of
; coincident spikes boths sets of data have and creates a uniform distribution
for the random data
; to see where the actual data lies.
; Defintion of variables:  yes_no_values is the binary spike train of the true
data
;                    random_array is the array of the location of spikes for the
simulated data
;


function count_random_matches_many, random_array, times_repeated, ge_or_eq_test,
window_size
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size

      random_matches = 0

      if (window_size eq 0) then begin
            for frames = 0, frame_no - 1 do begin
                  matches = total(random_array(*, frames))        ;finds number
of hits at this frame
                  if (((ge_or_eq_test eq 0) and (matches ge times_repeated)) or
((ge_or_eq_test eq 1) and (matches eq times_repeated))) then begin
                        random_matches = random_matches + 1
                  endif
            endfor
      endif else begin
            ;; first compress spike trains according to window size
            binned_random_array = intarr(n_elements(random_array[*,0]), frame_no
- (window_size * 2))
            for frame_counter = window_size, frame_no - (window_size + 1) do
begin
                  for window_counter = -window_size, window_size do begin
                        for cell_counter = 0, n_elements(random_array[*,0]) - 1
do begin
                              if (random_array[cell_counter, frame_counter +
window_counter] ge 1) then begin
                                    binned_random_array[cell_counter,
frame_counter - window_size] = 1
                              endif
                        endfor
                  endfor
            endfor
            ;; then look for matches in the binned spike trains (although a
single spike can fall into many bins)
            for frame_counter = 0, n_elements(binned_random_array[0, *]) - 1 do
begin
                  matches = total(binned_random_array(*, frame_counter))
;finds number of hits at this frame
                  if (((ge_or_eq_test eq 0) and (matches ge times_repeated)) or
((ge_or_eq_test eq 1) and (matches eq times_repeated))) then begin
```

```
                        random_matches = random_matches + 1
                    endif
                endfor
            endelse

            return, random_matches
end
```

```
; Name: delete_spikes_widget
;
; Description: allows me to delete from all cells a spike in a particular frame.

pro delete_spikes_widget
common delete_spikes, state12

      base12=WIDGET_BASE(/Column, title='Spike Deletion')
      button11=WIDGET_BUTTON(base12, VALUE='Delete All Trailing Spikes',
UVALUE=11)
      field12=CW_FIELD(base12, /RETURN_EVENTS, /INTEGER, TITLE='Frame for Global
Spike Deletion', VALUE=0, UVALUE=0)
      button12=WIDGET_BUTTON(base12, VALUE='Delete Spike in This Frame',
UVALUE=2)
      field13=CW_FIELD(base12, /RETURN_EVENTS, /FLOATING, TITLE='Threshold in
dF/F Units for Global Spike Deletion', VALUE=0.0, UVALUE=0)
      button13=WIDGET_BUTTON(base12, VALUE='Delete Spikes Less Than Threshold',
UVALUE=2)
      WIDGET_CONTROL, /realize, base12
      state12={button11:button11, field12:field12, button12:button12,
field13:field13, button13:button13}
      WIDGET_CONTROL, WIDGET_INFO(base12, /CHILD), SET_UVALUE=state12

      xmanager, 'delete_spikes_widget', base12
end
```

```
; Name: delete_spikes_widget_event
;
; Description: allows me to delete from all cells a spike in a particular frame.
;


pro delete_spikes_widget_event, event
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common delete_spikes, state12

      stateholder12=WIDGET_INFO(event.handler, /CHILD)
      WIDGET_CONTROL, stateholder12, GET_UVALUE=state12
      WIDGET_CONTROL, state12.field12, GET_VALUE=frame_to_kill
      WIDGET_CONTROL, state12.field13, GET_VALUE=spike_unit_threshold

      ;; deleting ALL trailing spikes in ALL cells
      if (event.id eq state12.button11) then begin
            for cell_counter = 0, (cell_no - 1) do begin
                  for frame_counter = (frame_no - 1), 2, -1 do begin
                        if ((yes_no_values[cell_counter, frame_counter] eq 1)
and ((yes_no_values[cell_counter, frame_counter - 1] eq 1) or
(yes_no_values[cell_counter, frame_counter - 2] eq 1))) then begin
```

```
                              yes_no_values[cell_counter, frame_counter] = 0
                        endif
                  endfor
            endfor
      endif

      ;; delete spike in every cell that is in a given frame
      if (event.id eq state12.button12) then begin
            if (frame_to_kill ge 0) and (frame_to_kill lt frame_no) then begin
                  for cell_counter = 0, (cell_no - 1) do begin
                        yes_no_values[cell_counter, frame_to_kill] = 0
                  endfor
            endif
      endif

      ;; delete all spikes less than a raw dF/F threshold of intensity
      if (event.id eq state12.button13) then begin
            for cell_counter = 0, (cell_no - 1) do begin
                  for frame_counter = 0, (frame_no - 1) do begin
                        if (pixel_array[cell_counter, frame_counter] lt
spike_unit_threshold) then begin
                              yes_no_values[cell_counter, frame_counter] = 0
                        endif
                  endfor
            endfor
      endif
end
```

```
; Name: draw_3D_plot
;
; Description: draws all frames of all cells, with intensity of dF/F on the z-
axis;
;     i.e. it makes a pretty cool plot!
;

pro draw_3D_plot, pixel_array

      window, 4, title='All Stocks-3D'
      surface, transpose(pixel_array), /horizontal
end
```

```
; NAME: draw_cgram
;
; INTRODUCTION:
;     This procedure is used to draw cross correlograms between any two cells. A
cross correlogram
; is a histogram of the time intervals between the two cells. If there is a peak
at 0, the cells are
; very highly correlated. This draws a cross correlogram between cell 2 with
respect to cell 1. That is
; a peak at +1 means that cell 2 leads cell 1 by 1 time frame.
;
; INPUTS:
;     pixel_array. This is the array created in either pixel_binary_cells or
pixel_binary_box
; depending on the choosing method you used. It contains the pixel values of the
cells and their
; converted binary form. It is the concated array of:
```

```
;
;
;                 pix_array=intarr(cell_no, frame_no)  {actually called
pixel_array in those pro}
;                 yes_no_values=intarr(cell_no, frame_no)
;
;     where cell_no is the number of cells you chose
;     and frame_no is the number of frames you chose to analyze
;     therefore, what we are talking about is:
;
;                 pixel_array =intarr(cell_no+cell+_no, frame_no)
;
;     to find the data on cell x,
;                 pixel_array(x-1,*) will give the pixel values over time (from
pix_values)
;                 pixel_array(cell_no+x-1,*) gives 1 for a peak else 0 (from
yes_no_values)
;
;     cell_number1: This is the number of the first cell you want to cross
correlate from
;     cell_number2: This is the number of the second cell you are considering
; binsize: This is the window size you are calculating with. If any spikes fall
in this, they are
; considered coincident. Default is 1
;     cell_no: Total number of cells you have chosen
;     time: The time between two frames shot by the CCD camera (input from
widget)
;
; DESCRIPTION:
;     This program looks at the yes_no_values part of pixel_array (look above)
and converts the
; binary array into an array holding the positions of the spikes. It uses the in
built function
; where() to do this. Then you use the function FIND_DISTANCE to calculate the
time differences between
; the spikes. It's source code is located in the same directory as this, 'All
you need'. Briefly what
; it does is it subtracts the two time arrays
;

pro DRAW_CGRAM, yes_no_values, cell_number1, cell_number2, bin_size, cell_no,
time, xmin, xmax

;this procedure will make a correlogram between the two files input.
;the functions it uses are:findgen(size)*bin_size+mintemp
;openfile, find_distance

!p.multi=0
forward_function FIND_DISTANCE
      write=string(format='("cross-correlogram of cell",/,(i0))', cell_number1)
      write2=string(format='(" vs cell ",/,(i0))', cell_number2)
      time1=where(yes_no_values(cell_number1-1, *) gt 0)    ;this finds where
the spikes are
      time2=where(yes_no_values(cell_number2-1, *) gt 0)
      IF ((time1(0) eq -1) OR (time2(0) eq -1)) then begin
            mess=WIDGET_MESSAGE('One of the cells has no spikes')
      ENDIF ELSE BEGIN
```

```
      distemp=FIND_DISTANCE(time1, time2)
      mintemp=min(distemp)
      maxtemp=max(distemp)
      xmin=float(xmin)/time
      xmax=float(xmax)/time
      size=xmax-xmin
      IF bin_size lt 1 THEN BEGIN
            mess=WIDGET_MESSAGE('Binsize must be >=1')
      ENDIF ELSE BEGIN
      hist=histogram(distemp, binsize=bin_size, min=xmin, max=xmax)
      plot, (findgen(size)*bin_size+xmin)*time, hist, psym=10,
title=write(0)+write(1)+write2(0)+write2(1),$
      xtitle='time in seconds.', ytitle='no of spikes'
      ENDELSE
      ENDELSE
end
```

```
pro DRAW_CGRAM2, yes_no_values, number1, number2, binsize, cell_no, time, xmin,
xmax

;this procedure will make a correlogram between the two files input.
;the functions it uses are:findgen(size)*bin_size+mintemp
;openfile, find_distance

!p.multi=0

      write=string(format='("cross-correlogram of cell",/,(i0))', number1)
      write2=string(format='(" vs cell ",/,(i0))', number2)
      time1=where(yes_no_values(number1 -1, *) gt 0)  ;this finds where the
spikes are
      time2=where(yes_no_values(number2 -1, *) gt 0)
      IF ((time1(0) eq -1) OR (time2(0) eq -1)) then begin
            mess=WIDGET_MESSAGE('One of the cells has no spikes')
      ENDIF ELSE BEGIN

      N_1=total(yes_no_values,2)
      N=N_1[number1 - 1]            ;this finds the total number of spikes for
cell number1

      sizeofyn_val=size(yes_no_values)
      actual_frame_no=sizeofyn_val[2]

      Si=intarr(actual_frame_no)
      for t=0, actual_frame_no - 1 do begin
      Si[t]=yes_no_values[number1 -1,t]
      endfor
      Sj=intarr(actual_frame_no)
      for t=0, actual_frame_no - 1 do begin
      Sj[t]=yes_no_values[number2 -1,t]
      endfor

      nlim=fix(- actual_frame_no / binsize) ;+1  ;be careful - some data is lost
when tha actual fr.# is not
      plim=fix(actual_frame_no / binsize) ;-1    ;be careful -divisible by
binsize
```

```
      ;print, 'nlim - lower bound =', nlim, '        plim - upper bound =', plim

      R=fltarr(2, plim - nlim +1)

      index=0
      for l=nlim, plim do begin
      z=0
      for t=0, actual_frame_no -1 do begin

      x=0
      for k=t+l*binsize, t+(l+1)*binsize -1 do begin   ;be careful with the -1
      if (k ge 0) and (k lt actual_frame_no) then begin
      x=x+Sj[k]
      endif
      endfor
      if(x gt 0) then begin
      x=1
      endif

      z=z + Si[t]*x  ;(x is either 0 or 1 depending whether cell j fired in a
given time bin)
      endfor
      R[0,index]=l
      R[1,index]=z
      index=index+1
      endfor

      for i=0, index -2 do begin
      R[1,i]=R[1,i]/(binsize * N)
      ;print, R[1,i]
      endfor


      center=-1
      for i=0, index-2 do begin
      if(R[0,i] eq 0) then center=i
      endfor

      xmin1=fix(xmin/time)
      xmax1=fix(xmax/time)

      IF binsize lt 1 THEN BEGIN
            mess=WIDGET_MESSAGE('Binsize must be >=1')
      ENDIF ELSE BEGIN

      IF (center + xmin1 lt 0) or (center + xmax1 gt index -1) then begin
            mess=Widget_Message('X is out of range')
      Endif else begin


      xaxis=[R[0,center + xmin1]*time,(INDGEN(- xmin1 + xmax1)
+R[0,center+xmin1] +1)*time]
      yaxis=fltarr(xmax1-xmin1 +1)
      for i=center + xmin1, center + xmax1 do begin
      yaxis[i - center - xmin1]=R[1,i]
      endfor
```

```
      plot, xaxis, yaxis , psym=10,
title=write(0)+write(1)+write2(0)+write2(1),$
      xtitle='time in seconds.', ytitle='firing rate'

      endelse
      ENDELSE
      ENDELSE
end
```

```
pro draw_cross
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov2, yes_no_values3, yes_no_temp  ;added by pvc3
      base=WIDGET_BASE(/column, title='Cross Correlogram')
      draw=WIDGET_DRAW(base, xsize=400, ysize=400)
      number1=CW_FIELD(base,title='Enter Cell Number 1', VALUE=1, UVALUE=2,$
            /RETURN_EVENTS, /INTEGER)
      number2=CW_FIELD(base, title='Enter Cell Number 2', VALUE=2, UVALUE=3,$
            /RETURN_EVENTS, /INTEGER)
      binsize=CW_FIELD(base, title='Enter the binsize', VALUE=1, UVALUE=4,$
            /RETURN_EVENTS, /INTEGER)
      xmin=CW_FIELD(base, title='x1=Lower bound', VALUE=-200, /RETURN_EVENTS,
/INTEGER)
      xmax=CW_FIELD(base, title='x2=Upper bound', VALUE=200, /RETURN_EVENTS,
/INTEGER)
      text=WIDGET_TEXT(base, VALUE=string('Total number of cells you chose:',
cell_no))
      WIDGET_CONTROL, base, /realize
      holder={number1:number1, number2:number2, binsize:binsize, draw:draw,
xmax:xmax, xmin:xmin}
      WIDGET_CONTROL, WIDGET_INFO(base, /Child), SET_UVALUE=holder
xmanager, 'draw_cross', base
end
```

```
pro draw_cross_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov2, yes_no_values3 ; added by pvc3
      stateholder5=WIDGET_INFO(event.handler, /Child)
      WIDGET_CONTROL, stateholder5, GET_UVALUE=state
      WIDGET_CONTROL, state.draw, GET_VALUE=win_id2
      WIDGET_CONTROL, state.number1, GET_VALUE=number_1
      WIDGET_CONTROL, state.number2, GET_VALUE=number_2
      WIDGET_CONTROL, state.binsize, GET_VALUE=bin_size
      WIDGET_CONTROL, state.xmax, GET_VALUE=xmax
      WIDGET_CONTROL, state.xmin, GET_VALUE=xmin

   IF ((number_1 gt cell_no) OR (number_1 lt 1)) OR ((number_2 gt cell_no) OR
(number_2 lt 1)) THEN BEGIN
            mess=WIDGET_MESSAGE(string('Invalid cell number entered. You only
chose',byte(cell_no),' cells'), /ERROR)
    ENDIF ELSE BEGIN
```

```
      wset, win_id2      ;changed by pvc3

      draw_cgram, yes_no_values3, number_1, number_2, bin_size, cell_no,
time_resolution, $
                  xmin, xmax
    ENDELSE
 end
```

```
pro draw_cross2
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov2, yes_no_values3, yes_no_temp  ;added by pvc3
      base=WIDGET_BASE(/column, title='Cell firing rate Cross Correlogram')
      draw=WIDGET_DRAW(base, xsize=400, ysize=400)
      number1=CW_FIELD(base,title='Enter Cell Number 1', VALUE=1, UVALUE=2,$
            /RETURN_EVENTS, /INTEGER)
      number2=CW_FIELD(base, title='Enter Cell Number 2', VALUE=2, UVALUE=3,$
            /RETURN_EVENTS, /INTEGER)
      binsize=CW_FIELD(base, title='Enter the binsize', VALUE=1, UVALUE=4,$
            /RETURN_EVENTS, /INTEGER)
      xmin=CW_FIELD(base, title='x1=Lower bound', VALUE=-200, /RETURN_EVENTS,
/INTEGER)
      xmax=CW_FIELD(base, title='x2=Upper bound', VALUE=200, /RETURN_EVENTS,
/INTEGER)
      text=WIDGET_TEXT(base, VALUE=string('Total number of cells you chose:',
cell_no))
      WIDGET_CONTROL, base, /realize
      holder={number1:number1, number2:number2, binsize:binsize, draw:draw,
xmax:xmax, xmin:xmin}
      WIDGET_CONTROL, WIDGET_INFO(base, /Child), SET_UVALUE=holder
xmanager, 'draw_cross2', base
end
```

```
pro draw_cross2_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov2, yes_no_values3 ; added by pvc3
      stateholder5=WIDGET_INFO(event.handler, /Child)
      WIDGET_CONTROL, stateholder5, GET_UVALUE=state
      WIDGET_CONTROL, state.draw, GET_VALUE=win_id2
      WIDGET_CONTROL, state.number1, GET_VALUE=number_1
      WIDGET_CONTROL, state.number2, GET_VALUE=number_2
      WIDGET_CONTROL, state.binsize, GET_VALUE=bin_size
      WIDGET_CONTROL, state.xmax, GET_VALUE=xmax
      WIDGET_CONTROL, state.xmin, GET_VALUE=xmin

   IF ((number_1 gt cell_no) OR (number_1 lt 1)) OR ((number_2 gt cell_no) OR
(number_2 lt 1)) THEN BEGIN
            mess=WIDGET_MESSAGE(string('Invalid cell number entered. You only
chose',byte(cell_no),' cells'), /ERROR)
    ENDIF ELSE BEGIN
```

```
      wset, win_id2      ;changed by pvc3

      draw_cgram2, yes_no_values3, number_1, number_2, bin_size, cell_no,
time_resolution, $
                  xmin, xmax
    ENDELSE
 end
```

```
; Name: draw_raster
; Description: This will draw spikes for those points that pass the test.
; Working: It sees if the pixel vs time graph has any peaks and then simply
draws a line
; from the x-axis to represent an 'on' or spike.
; It draws the plots one on top of another in a raster form.

pro draw_raster
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common choose_cell_convert_delta, filepath1

      a=bytarr(frame_no+1)
      loadct, 27
      !p.multi=[0, 1, cell_no]
      window, 1, xsize=600, ysize=600,title=strcompress(strmid(systime(), 3, 8))

      ; Hack to print out raster values to file
      filename=pickfile(/write, file='Location_Spikes_#_')
      if (filename eq '') then begin
            MESS=WIDGET_MESSAGE('Data will not be saved!', /INFORMATION)
            cd, ''
            filename='rasterdata.txt'
      endif

      openw, outunit, filename, /GET_LUN
      printf, outunit, FORMAT= '("Number of Frames: ",I0,"    Number of Cells:
",I0)',frame_no,cell_no
      printf, outunit, ''
      printf, outunit, 'Cell No         No Spikes     Spike Times (frame)'

      for cell_number = 0, cell_no - 1 do begin
            if (cell_number le 8) then begin  ;; 8 because we add one to the
cell number when we print
                  printf, outunit, FORMAT= '("Cell  ",I0,":",I,"    ",$)',
cell_number+1, total(yes_no_values(cell_number,*))
            endif else begin
                  printf, outunit, FORMAT= '("Cell ",I0,":",I,"    ",$)',
cell_number+1, total(yes_no_values(cell_number,*))
            endelse

            ;; code to write out only those frames that have spikes
            if (total(yes_no_values(cell_number,*)) ne 0) then begin
      ;this only draws cells that have activity
                  plot, a, xstyle=4, ystyle=4, ymargin=[0,0], xmargin=[15,0]
;xstyle and ystyle avoid display of x and y axis respec. see 2-8R
                  xyouts, -(frame_no/10.000), 0, cell_number+1, charsize=0.8
      ;this prints the cell numbers in front of the cells
```

```
                for frames=0, frame_no-1 do begin
                      if (yes_no_values(cell_number, frames) ge 1) then begin
;to read yes_no
                            plots, [frames, frames], [.75, 0], color=3
                            printf, outunit, FORMAT= '($,I)', frames
                      endif
                endfor
          endif

          ;; code to write yes_no_values instead- i.e. writes the whole binary
spike train for each cell
          ;printf, outunit, ''
          ;printf, outunit, transpose(yes_no_values[cell_number,*])

          printf, outunit, ''  ; newline
     endfor

     CLOSE, outunit
     FREE_LUN, outunit
     !p.multi=0
end
; NAME: draw_significance_raster.pro
;
; DESCRIPTION:  This program looks at the many_one and two_many significance
tests to see
;     whether many cells fire in one frame or a pair of cells fire together in
many frames.
;     It then colors the two cases differently on the raster plot.
;     This says nothing about the significance of the correlation.  For that,
you must look at the
;     distribution curve or the data file that pops up after calling many_one or
two_many
;

pro draw_significance_raster, yes_no_significance

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
     total_frame_no, time_resolution, x_size, y_size, box_size
common with_create_dist, cells_active                ;cells_active is the
number of active cells in the movie

     ;_____________________DRAW THE BASE LINES FOR THE CELLS AND PLOT THE
SPIKES ON THEM_____________________________
     a = bytarr(frame_no + 1)            ;plotting 'a' will make the line on
which the spikes will be for each cell
     loadct, 27
     !p.multi=[0, 1, cells_active] ;this plots cells_active many single lines
one above another
     window, /free, xsize=600, ysize=600,title='Significance Raster Plot'
          ;the title is the date

     for cell_number = 0, cell_no - 1 do begin
        if (total(yes_no_significance(cell_number,*)) ne 0) then begin
;this only prints cells with activity
           plot, a, xstyle=4, ystyle=4, ymargin=[0,0], xmargin=[15,0]
          ;xstyle and ystyle avoid display of x and y axis respec. see 2-8R
```

```
                xyouts, -(frame_no / 10.000), 0, cell_number + 1, color=2
      ;this prints the cell numbers in front of the cells
                for frames = 0, frame_no - 1 do begin
      ;_____________________________COLOR THE SPIKES RED IF THEY HAPPEN TO FULFILL
THE CRITERIA_____________________
                        if (yes_no_significance(cell_number, frames) ge 1) then
begin ;to read yes_no
                                plots, [frames, frames], [.75, 0],
color=(yes_no_significance(cell_number,frames)-1)*200+2     ;this makes the
correlated spikes red and others grey
                        endif

                endfor
           endif
      endfor

      ;__________________PRINTING A DOTTED BLUE LINE OVER THE MANY CELLS THAT FIRE
TOGETHER IN ONE FRAME_____________________
      for frames = 0, frame_no - 1 do begin
            if total(yes_no_significance(*, frames) gt 1) gt 0 then begin
            plots, [frames, frames], [200,0], color=3, linestyle=1
            endif
      endfor

      !p.multi=0
end
```

```
; Name: draw_spikes
; Description: This will draw spikes for those points that pass the test.
; Working: It sees if the pixel vs time graph has any peaks and then simply
draws a line
; from the x-axis to represent an 'on' or spike.

pro draw_spikes, number_1

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size

common share_w_pixel_vs_time_widget_com, y_min, y_max, spike_min, spike_max,
x_max, x_min

!p.multi=0

plot, findgen(frame_no)*time_resolution, pixel_array(number_1-1, *),
yrange=[y_min, y_max],$
            xrange=[x_min, x_max], title=number_1, ytitle='% Change in Index',
xtitle='Time (days)', FONT=-1
      for frames=1, frame_no-1 do begin         ;this doesn't consider a spike
at the 1st frame
            if (yes_no_values(number_1-1, frames) ne 0) then begin
                  plots, [frames*time_resolution, frames*time_resolution],
[spike_max, spike_min]
            endif
      endfor
end
```

```
; Name: draw_spikes_many_cell_delta
```

```
; Synopsis:
; Description: This program plots the deltaF/F values of a single cell per time.

pro draw_spikes_many_cell_delta, delta_f_values, cell_number,
frame_number,time_resolution

      !p.multi=0
      window, 2, title='Superimposed Plots'

      plot, findgen(frame_number)*time_resolution, delta_f_values(0, *),
yrange=[min(delta_f_values),max(delta_f_values)],$
            title='All Stocks', ytitle='% Change in Index', xtitle='Time (days)'

      xyouts, -6, 0, 1
      if (cell_number gt 1) then begin
            for cell_numb=1, cell_number-1 do begin
                  oplot, findgen(frame_number)*time_resolution,
(delta_f_values(cell_numb,*))
                  a=(delta_f_values(cell_numb,*) eq
max((delta_f_values(cell_numb,*))))
                  a=where(a eq 1)
                  xyouts, a,max((delta_f_values(cell_numb,*))), cell_numb+1
            endfor
      endif
end
```

```
; Name: draw_spikes_many_pages

pro draw_spikes_many_pages, pixel_array, yes_no_values, cell_no,
frame_number,time_resolution

common share_w_pixel_vs_time_widget_com, y_min, y_max, spike_min, spike_max


sum_array=intarr(cell_no)                              ;find which cells are
active
for counter=0, cell_no-1 do begin
      sum_array(counter)=total(yes_no_values(counter,*))
endfor
active_cells=total(sum_array gt 0)                ;this has the location of the
cells that are active
if active_cells eq 0 then begin
      mess=WIDGET_MESSAGE('There are no active stocks', /error)
endif else begin
      active_cell_subscripts=where(sum_array gt 0)                ;this is equal
to the number of active cells

      number = (active_cells / 9) + 1                                    ;number
of windows that have to be opened
      for t = 0, number - 1 do begin                                ;open the
windows
            window, t, xsize=500, ysize=500
      endfor

      !p.multi=[0,3,3]                                    ;it'll print three rows of
three cells

      for index=0, active_cells-1 do begin
```

```
            wset, (index) / 9
            cell_number=active_cell_subscripts(index)
            plot, findgen(frame_number)*time_resolution,
pixel_array(cell_number, *), yrange=[y_min, y_max],$
            title=cell_number+1, ytitle='% Change in Index', xtitle='Time
(days)'

            for frames=0, frame_number-1 do begin
                  if (yes_no_values(cell_number, frames) ge 1) then begin
                        plots, [frames*time_resolution, frames*time_resolution],
[spike_max, spike_min]
                  endif
            endfor
      endfor
endelse
!p.multi=0                                          ;this returns plotting to
systems default
end
```

```
; Name: filter_median_subtractive
;
;; Parameters: takes an input array (the array of cells x frames dF/F values), a
window size for
;     filtering, and an output array (an empty variable which will be the
filtered input passed back).
;
;

pro filter_median_subtractive, input_array, median_window_size, output_array

      output_array = float(input_array)
      num_cells = n_elements(input_array(*,0))
      num_frames = n_elements(input_array(0,*))
      temp_array = fltarr(num_frames + (2 * median_window_size))

      for i = 0, (num_cells - 1) do begin
            temp_array(median_window_size:median_window_size + num_frames - 1) =
reform(input_array(i,*))
            temp_array = (temp_array - median(temp_array, median_window_size))
            output_array(i,*) = temp_array(median_window_size:median_window_size
+ num_frames - 1)
      endfor
end
```

```
; Name: find_distance
; Description: This procedure is used by draw_cgram to make cross correlograms
and auto correlograms.
; Timearray1 and timearray2 are arrays which hold the times where the spikes
occur. The output
; of this procedure is an array of the difference of the second spike train from
the first cell's.
; It does this by taking the spike of the first cell eg 13 54 66 154 and looks
at the second spike
; train's timearray which may look like: 14 55 67 155
; outputarray(0, 0)=14-13=1
; outputarray(0, 1)=14-54=-40
; outputarray(0, 2)=14-66=-52
```

```
; outputarray(0, 3)=14-154=-140
; outputarray(1, 0)=55-13=42
; outputarray(1, 1)=55-54 =1
; and so on
; this will show a peak at 1 in the cross correlogram for 1 to 2.


function find_distance, timearray1, timearray2
;this function takes two files and finds how their elements differ by
subtracting all of the elements
; of one from the other. Output is a file with a measure of their deviation from
each other

      size1=n_elements(timearray1)
      size2=n_elements(timearray2)
      outputarray=intarr(size1, size2)
      index=0
      for i=0, size1-1 do begin          ;one by one subtracting the elements
of file1
                                    ;from the whole of file2
            outputarray(index, *)=timearray2-timearray1(i)
            index=temporary(index)+1
      endfor
      ;for j=0, size1-1 do begin
            ;for t=0, size2-1 do begin
                  ;outputarray(j,t)=timarray2(t)-timearray1(j)
            ;endfor
      ;endfor
return, outputarray
end
```

```
; NAME: find_matches.pro
;
; INTRODUCTION:
;     This calculates the correlation coefficient of two cells with a binsize
;
; INPUTS:
;     cell1 and cell2 are the unidimensional arrays of the timearrays of the
cells.
;       winsize is the size of the window. THIS MUST BE AN ODD NUMBER.
;
; DESCRIPTION:
;      The c coeff is calculated as follows.
;           1. The binary spikes are converted into time arrays where the values
of the
; array correspond to the indices where the spikes occurred.
;           For example, if cell1=[0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1]
;                       time1=where(cell1 gt 0)
;                       IDL> print, where(cell1 gt 0)
;                          4           7          10          11
;           2. The IDL library function strpos is used to find whether any of
the time array
; values are the same. If there are two time array values that are the same,
that means that both of
; the cells we are analyzing had spikes at the same time. That counts as a match
in the formula to
; calculate correlation coefficients:
;            c_coeff= number of matches(A->B)/ total spikes of A
```

```
;
;           3. find_matches counts the number of times there are congruent spike
positions.  The
; number of these occurences = matches
;
;           4. The total number of spikes in cell 1 =length1
;
;           5 Cross correlation coef=matches/length1

function find_matches, cell1, cell2, winsize

matches=0                       ; matches is the number of coincident spikes
time1=where(cell1 gt 0)               ;gives the location of where cell1 is
greater than 0
time2=where(cell2 gt 0)               ;gt means greater than. similarly->ge is
greater than equal
length1=total(time1 gt 0)             ;length1 gives the number of elements in the
array time1
                                ;which is also the number of spikes in cell1
                                ;the same window (whose size is defined by
winsize) as a spike
                                ;from cell2. It is calculated in the module below


;_____________________FIND THE NUMBER OF COINCIDENT
SPIKES___________________________________

for i=0, length1-1 do begin
      window_size=(winsize-1)/2      ;a window is created x before a spike and x
after. Therefore making it 2x+1 large
      lower_window_limit=time1(i)-window_size
      upper_window_limit=time1(i)+window_size
      count=0                        ;count is the number of times a spike from
cell1 occurs in
      for m=lower_window_limit, upper_window_limit do begin
            temp=strpos(time2, m)
            temp1=(temp)+1           ;there is a zero if there is a match
therefore we have to add 1
            if (total(temp1) ne 0) then count=count+1   ;in this case there is a
coincident spike
      endfor
      if (count gt 0) then matches=matches+1
endfor
;___________________CALCULATE THE CORRELATION
COEFFICIENT_____________________________________

coeff=matches/(length1)             ;length1 is always greater than 0 as we only
do this for active cells
return, [coeff]
end
```

```
; gen_sig_widget

pro gen_sig_widget

      gen_sig_base=WIDGET_BASE(/COLUMN, title='General Significance')
```

```
      gen_sig_num_trials=CW_FIELD(gen_sig_base, /RETURN_EVENTS, /INTEGER,
TITLE='Number of random trials to run:', VALUE=1000, UVALUE=0)
      gen_sig_window_size=CW_FIELD(gen_sig_base, /RETURN_EVENTS, /INTEGER,
TITLE='Window size:', VALUE=0, UVALUE=0)
      gen_sig_button=WIDGET_BUTTON(gen_sig_base, VALUE='Find significance',
UVALUE=2)

      WIDGET_CONTROL, /realize, gen_sig_base
      gen_sig_state={gen_sig_num_trials:gen_sig_num_trials,$
                     gen_sig_window_size:gen_sig_window_size,$
                     gen_sig_button:gen_sig_button}
      WIDGET_CONTROL, WIDGET_INFO(gen_sig_base, /CHILD),
SET_UVALUE=gen_sig_state
      xmanager, 'gen_sig_widget', gen_sig_base
end
;-------------------------------------------------------------------------
;     Name: gen_sig_widget_event
;

pro gen_sig_widget_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_create_dist, cells_active

      gen_sig_stateholder=WIDGET_INFO(event.handler, /CHILD)
      WIDGET_CONTROL, gen_sig_stateholder, GET_UVALUE=gen_sig_state
      WIDGET_CONTROL, gen_sig_state.gen_sig_num_trials, GET_VALUE=no_of_times
      WIDGET_CONTROL, gen_sig_state.gen_sig_window_size, GET_VALUE=window_size

      if (event.id eq gen_sig_state.gen_sig_button) then begin
            if ((no_of_times le 1) or (window_size le -1)) then begin
                  error_dialog = WIDGET_MESSAGE('Invalid fields specified!',
/error)
            endif else begin
                  ;; We write all of our data into the following file in the
function find_general_p
                  filename3 = 'General_Statistical_Data'
                  filename3 = pickfile(/write, file='General_Stats_#')
                  if (filename3 eq '') then begin
                        mess = WIDGET_MESSAGE('The Data will not be saved.',
/INFORMATION)
                        filename3 = 'General_Statistical_Data'
                  endif

                  WIDGET_CONTROL, /HOURGLASS

                  ;; creating the random distribution
                  times_repeated = 2
                  no_spikes = total(yes_no_values)
                  distribution_array = intarr(no_of_times) ;; array of hits for
each random trial
                  for t = 0, no_of_times - 1 do begin
                        make_random_data, seed, random_array, num_active_cells
                        distribution_array(t) =
count_random_matches(random_array, times_repeated, window_size)
                  endfor
```

```
                    ;; count the number of matches in the real data now
                    true_matches = 0
                    if (window_size eq 0) then begin
                          for frame_counter = 0, frame_no - 1 do begin
                                matches = total(yes_no_values(*, frame_counter))
;this gives us the n in our combination formula above
                                if (matches ge times_repeated) then begin
                                      matches_pair = factorial(matches) /
(factorial(times_repeated) * factorial(matches - times_repeated))    ; finds
number of possible pairs
                                endif else begin
                                      matches_pair = 0                    ;cant find
factorial of a negative number
                                endelse
                                true_matches = true_matches + matches_pair
;increases the matches count for the true data by that of this frame
                          endfor
                    endif else begin
                          ;; first compress spike trains according to window size
                          binned_yn_vals_array = intarr(cell_no, frame_no -
(window_size * 2))
                          for frame_counter = window_size, frame_no - (window_size
+ 1) do begin
                                for window_counter = -window_size, window_size do
begin
                                      for cell_counter = 0, cell_no - 1 do begin
                                            if (yes_no_values[cell_counter,
frame_counter + window_counter] ge 1) then begin

      binned_yn_vals_array[cell_counter, frame_counter - window_size] = 1
                                            endif
                                      endfor
                                endfor
                          endfor
                          ;; then look for matches in the binned spike trains
(although a single spike can fall into many bins)
                          for frame_counter = 0,
n_elements(binned_yn_vals_array[0, *]) - 1 do begin
                                matches = total(binned_yn_vals_array(*,
frame_counter))        ;finds number of hits at this frame
                                if (matches ge times_repeated) then begin
                                      matches_pair = factorial(matches) /
(factorial(times_repeated) * factorial(matches - times_repeated))    ; finds
number of possible pairs
                                endif else begin
                                      matches_pair = 0                    ;cant find
factorial of a negative number
                                endelse
                                true_matches = true_matches + matches_pair
;increases the matches count for the true data by that of this frame
                          endfor
                    endelse

                    ;; determine connectivity stats
                    distances_array = [0.0]
                    total_connections  = 0
```

```
                    connections_array = intarr(cell_no)
                    if (window_size eq 0) then begin
                          for cell_1 = 0, cell_no - 2 do begin
                                for cell_2 = cell_1 + 1, cell_no - 1 do begin
                                      temp_cell = yes_no_values(cell_1, *) *
yes_no_values(cell_2, *)
                                      if ((total(temp_cell)) ge 1) then begin
                                            connections_array[cell_1] =
connections_array[cell_1] + 1
                                            connections_array[cell_2] =
connections_array[cell_2] + 1
                                            total_connections = total_connections
+ 1 ;; do this so a connection is only counted once, if it matters
                                            connection_distance =
sqrt(((double(location(cell_1).coord[0]) - double(location(cell_2).coord[0])) ^
2) + ((double(location(cell_1).coord[1]) - double(location(cell_2).coord[1])) ^
2))
                                            distances_array =
[distances_array,connection_distance]
                                      endif
                                endfor
                          endfor
                    endif else begin
                          for cell_1 = 0, cell_no - 2 do begin
                                for cell_2 = cell_1 + 1, cell_no - 1 do begin
                                      temp_cell = binned_yn_vals_array(cell_1, *)
* binned_yn_vals_array(cell_2, *)
                                      if ((total(temp_cell)) ge 1) then begin
                                            connections_array[cell_1] =
connections_array[cell_1] + 1
                                            connections_array[cell_2] =
connections_array[cell_2] + 1
                                            total_connections = total_connections
+ 1 ;; do this so a connection is only counted once, if it matters
                                            connection_distance =
sqrt(((double(location(cell_1).coord[0]) - double(location(cell_2).coord[0])) ^
2) + ((double(location(cell_1).coord[1]) - double(location(cell_2).coord[1])) ^
2))
                                            distances_array =
[distances_array,connection_distance]
                                      endif
                                endfor
                          endfor
                    endelse

                    ;; Begin printing the data
                    close,  1
                    openw, 1, filename3
                    printf, 1, 'STATISTICAL DATA'
                    printf, 1, ' '
                    printf, 1, 'Random Test Information'
                    printf, 1, '-----------------------'
                    printf, 1, 'How many times a pair is together:  ',
strcompress(times_repeated)
                    printf, 1, 'Number of iterations in tests:            ',
strcompress(no_of_times)
```

```
                printf, 1, 'Window size:                                    ',
strcompress(window_size)
                printf, 1, 'Total number of active stocks:                  ',
strcompress(round(cells_active))
                printf, 1, ' '

                if (max(distribution_array) eq 0) then begin
                     mess = WIDGET_MESSAGE('Two cells never fire together
randomly!  Try the MANY CELLS AT ONCE option looking at 2 cells firing together
with more than 100 iterations.', /error)
                     printf, 1, 'ERROR!'
                     printf, 1, 'Two cells never fire together randomly.
Nothing can be said about the significance of this data'
                     real_p = -1
                endif else begin
                     loadct, 27
                     hist = histogram(distribution_array, binsize=1, min=0,
max=(max([true_matches,distribution_array]) + 2))
                     window, /free, title='Distribution for General
Significance'
                     plot, hist, xtitle='number of hits', ytitle='frequency'
                     plots, [true_matches, true_matches], [0,
max([1,total(distribution_array eq true_matches)])], color=12     ;this draws a
line where the actual number of matches lies

                     spikes_per_cell_per_second = (no_spikes / cells_active)
/ (frame_no * time_resolution)

                     stats = moment(distribution_array, sdev=gen_sdev)
                     p_value = gauss_pdf((true_matches - stats(0)) /
gen_sdev)
                     real_p = 1 - p_value

                     printf, 1, 'General Statistical Information'
                     printf, 1, '-------------------------------'
                     printf, 1, 'Total number of frames:       ',
strcompress(frame_no)
                     printf, 1, 'Time between frames:          ',
strcompress(time_resolution)
                     printf, 1, 'Total number of stocks:       ',
strcompress(cell_no)
                     printf, 1, 'Total number of spikes:       ',
strcompress(no_spikes)
                     printf, 1, 'Mean expected matches:        ',
strcompress(stats(0))
                     printf, 1, 'Variance:                     ',
strcompress(stats(1))
                     printf, 1, 'Standard deviation:           ',
strcompress(gen_sdev)
                     printf, 1, 'Actual matches:               ',
strcompress(true_matches)
                     printf, 1, 'Actual/expected:              ',
strcompress(true_matches / stats(0))
                     printf, 1, 'Standard error for ratio:           ',
strcompress(gen_sdev / stats(0))
                     printf, 1, 'Mean spikes per stock:        ',
strcompress(no_spikes / cells_active)
```

```
                        printf, 1, 'Spike firing rate:                    ',
strcompress(spikes_per_cell_per_second)
                        printf, 1, 'Significance p-value:          ',
strcompress(real_p)
                        printf, 1, '(IMPORTANT! If this is exactly .5, it could
be actually very significant so look at actual/expected!)'
                        printf, 1, ' '

                        printf, 1, 'Stock Connectivity Information'
                        printf, 1, '------------------------------'
                        printf, 1, 'Total Number of Connections:                ',
strcompress(total_connections)

                        connected_cells = where(connections_array)
                        num_connected_cells = n_elements(connected_cells)
                        if (n_elements(connected_cells) ge 2) then begin
                              stats_conn_only =
moment(connections_array(connected_cells), sdev=sdev_conn_only)
                              mean_connections = stats_conn_only[0]
                              normalized_slice_connectivity = (mean_connections
/ num_connected_cells)
                              printf, 1, 'Mean Number of Connections per Stock:
      ', strcompress(mean_connections)
                              printf, 1, 'Standard Deviation:
            ', strcompress(sdev_conn_only)
                        endif else begin
                              normalized_slice_connectivity = 0.0 ;; no
connected cells! Can't have just one connected cell.
                        endelse
                        printf, 1, 'Normalized Connectivity:
      ', strcompress(normalized_slice_connectivity)
                        printf, 1, 'Number of Silent Stocks:
      ', strcompress(cell_no - num_connected_cells)
                        printf, 1, 'Number of Coactive Stocks:
      ', strcompress(n_elements(connected_cells))
                        printf, 1, 'Minimum Number of Connections:
      ', strcompress(min(connections_array))
                        printf, 1, 'Maximum Number of Connections:
      ', strcompress(max(connections_array))

                        if (n_elements(distances_array) gt 2) then begin
                              distance_stats =
moment(distances_array[1:n_elements(distances_array) - 1],
sdev=sdev_distances_array)
                              printf, 1, 'Mean Connection Distance:
            ', strcompress(distance_stats[0])
                              printf, 1, 'Standard Deviation:
            ', strcompress(sdev_distances_array)
                        endif
                  endelse

                  ;; Close the file we are writing, reopen it if small enough
                  close, 1
                  openr, 1, filename3
                  stat_stuff = fstat(1)
                  file_size = stat_stuff.size
                  close, 1
```

```
                  if (file_size gt 8112) then begin
                        mess=WIDGET_MESSAGE('File is too large to display
through a widget.  Open it manually.  If you did not save it, it is named
General_Statistical_Data', /INFORMATION)
                  endif else begin
                        xdisplayfile, filename3, title = "General Statistical
Data", group = event.top, width = 75, height = 50
                  endelse
            endelse
      endif
end
```

```
;; This file reads MS Excel file with stock data of the following format
;; date     ticker1     ticker2     ...   ...
;; value    close1      close2      ...   ...
;;    ..    ..          ..          ..    ..
;; This file creates the following arrays:
;; symbol_array (string) - first row of the excle file minus first value
;; date_array (long or string or date) - first column of the excel file minus
the first value
;; pixel_array (float) - all the rest
;; pixel_array later gets transformed by calculating deltaF/F - under the same
name


pro load_and_convert_excelfile

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common test, str_ing, state3
common old_skool_data, original_data
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common textfile_vars, text_flag, diode_array, max_num_of_diodes
common stockdata, symbol_array, date_array


      message_dialog=WIDGET_MESSAGE("This procedure loads stock data, it creates
symbol_array, date_array, and pixel_array", /INFORMATION)

      filename=dialog_pickfile(/read, file=('stocks.slk'), get_path=filepath1)
      if (filename eq '') then begin
            message_dialog=WIDGET_MESSAGE("No data read.", /INFORMATION)
      endif else begin
            close, /all
            openr, 1, filename


            ;use excel import function to fill all the required arrays
            symbol_array= read_sylk(filename, /ARRAY,nrows=1, startcol=1)
            date_array=read_sylk(filename, /ARRAY, ncols=1,
startrow=1,/uselongs)
            data = read_sylk(filename, /ARRAY, startrow=1, startcol=1)
            close, /all

            cell_no = n_elements(symbol_array)
```

```
            frame_no = n_elements(date_array)

            WIDGET_CONTROL, state3.name, SET_VALUE=strmid(filename,
strlen(filepath1))
            WIDGET_CONTROL, state3.frame, SET_VALUE=frame_no
            WIDGET_CONTROL, state3.time, SET_VALUE=time_resolution

            ;; build 'pixel_array'
            pixel_array = fltarr(cell_no, frame_no)
            for j=0, frame_no -1 do begin
                  for i = 0, cell_no - 1 do begin
                        pixel_array[i,j] = data[j,i]
                  endfor
            endfor
            original_data = pixel_array
            ;the following finds delta F over F
            deltaf=fltarr(cell_no, frame_no)

            for i = 0, cell_no - 1 do begin
                  for j=1, frame_no-1 do begin
                        deltaf[i,j]=100*(pixel_array[i,j] - pixel_array[i,j-
1])/pixel_array[i,j-1]
                  endfor
            endfor
            for i = 0, cell_no - 1 do begin
                  deltaf[i,0] = 0
            endfor

            pixel_array=deltaf


            ;; initialize other variables
            yes_no_values = intarr(cell_no, frame_no)
            coef = fltarr(cell_no, cell_no)
            rms_threshold = fltarr(cell_no)
            threshold1 = fltarr(cell_no)
            threshold2 = fltarr(cell_no)
            for cell = 0, cell_no - 1 do begin
                  rms_threshold[cell] = 2.0
                  threshold1[cell] = 2.0
                  threshold2[cell] = 3.0
            endfor
            cells_defined = 1
            spikes_defined = 0
            single_plot_defined = 0
            correl_coef_defined = 0
            box_size = 2
            x_size = 350
            y_size = 350
            ;; locations are assigned deterministically around a circle
            location = replicate({struct, coord:intarr(2), size:0,
half_side:0.00}, cell_no)
            for i = 0, (cell_no - 1) do begin
                  location(i).size = cell_no
                  location(i).half_side = 1
                  location(i).coord[0] = cell_no * 5 + 1
                  location(i).coord[1] = cell_no * 5 + 1
```

```
            endfor

            text_flag = 1

            message_dialog=WIDGET_MESSAGE("Finished reading data.",
/INFORMATION)
      endelse
end
```

```
; Name: make_binary

function make_binary, delta_values, frame_no, cell_no, threshold1, threshold2

      binary_values = intarr(cell_no, frame_no)

      for cell = 0, cell_no - 1 do begin
            Case 1 of
            (threshold1[cell] gt 0): begin                  ;for increasing
spikes, the threshold will be greater than 0
                  for frame=1, frame_no-1 do begin    ;it doesn't consider a
spike at the first frame
                        if (frame lt frame_no-2) then begin       ;for all of
the peaks upto the last frame
                              if (((delta_values(cell, frame+1))-
(delta_values(cell, frame)) ge threshold1[cell]) OR $
                                    (delta_values(cell, frame+2)-delta_values(cell,
frame) ge threshold2[cell])) then begin
                                    binary_values(cell, frame)=1
                              endif
                        endif else begin  ;this takes care of a peak at last
frame
                              if ((delta_values(cell, frame_no-1))-
(delta_values(cell, frame_no-2)) ge threshold1[cell]) then begin
                                    binary_values(cell, frame_no-2)=1
                              endif
                        endelse
                  endfor
            end

            (threshold1[cell] lt 0): begin                  ;for decreasing
spikes, the threshold will be less than 0
                  for frame=1, frame_no-1 do begin    ;it doesn't consider a
spike at the first frame
                        if (frame lt frame_no-2) then begin
                              if (((delta_values(cell, frame+1))-
(delta_values(cell, frame)) le threshold1[cell]) OR $
                                    (delta_values(cell, frame+2)-delta_values(cell,
frame) le threshold2[cell])) then begin
                                    binary_values(cell, frame)=1
                              endif
                        endif else begin  ;this takes care of a peak at last
frame
                              if ((delta_values(cell, frame_no-1))-
(delta_values(cell,frame_no-2)) le threshold1[cell]) then begin
                                    binary_values(cell, frame_no-2)=1
                              endif
                        endelse
                  endfor
```

```
            end
            endcase
      endfor

return, binary_values
end
```

```
; Name: make_random_data.pro
; Description: This program looks at the input cell's activity and creates a
random set of data
; based on this data, by rotating each cell's spike train by a random amount.
;     Please note that this random array contains ONLY those cells which spike
at least once!

pro make_random_data, my_seed, random_array, num_active_cells

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_create_dist, cells_active

      ;find the number of spikes each cell of the true data has
      no_spikes = intarr(cell_no)                                    ;this is the
array with the number of spikes for each cell
      for cell = 0, cell_no - 1 do begin
            no_spikes(cell) = total(yes_no_values(cell, *))
      endfor

      cells_active = total(no_spikes gt 0)
      num_active_cells = cells_active
      random_array = intarr(cells_active, frame_no)
      random_cell_index = 0

      for cell_counter = 0, cell_no - 1 do begin
            if (no_spikes(cell_counter) ge 1) then begin
                  ;; pick a random int between 0 and frame_no
                  random_time_shift = fix(randomu(my_seed) * frame_no)
                  random_array[random_cell_index,*] =
shift(yes_no_values[cell_counter,*], random_time_shift)
                  random_cell_index = random_cell_index + 1
            endif
      endfor
end
```

```
; Name:
;     make_single_binary
; Description:
;     This program comes under the single plot call. It is called when the user
enters 2 thresholds that
; are different from the 2 thresholds that calculated the spikes for the whole
movie (under the choose threshold call).
; Make_binary calculated the spikes for the whole movie while this function is
called make_single_binary.
; This calculates the spikes only when told to do so for a single cell under the
single plot widget.  Therefore it is called
; make_single_binary.
```

```
;     It goes through the deltaF/F values (delta_values) and sees if the
absolute increase in delta_value between adjacent frames
; is greater than or equal to the threshold for positive spikes and less than or
equal to the threshold for negative spikes.
; We do not consider spikes at the first frame.
;
; Explanation of variables:
;     delta_values: the array with the actual delta F/F values
;     yes_no_values:  the converted binary array.
;     cell_number: the cell whose spikes are being found.
;     frame_no: total number of frames
;     thresh1: threshold between two adjacent frames
;     thresh2: threshold between three adjacent frames
;

function make_single_binary, delta_values, yes_no_values, cell_number, frame_no,
thresh1, thresh2

yes_no_values(cell_number-1, *)=0
Case 1 of
(thresh1 gt 0): begin                        ;for increasing spikes, the threshold
will be greater than 0
      for frame=1, frame_no-1 do begin       ;this doesn't consider a spike at the
first frame
            if (frame lt frame_no-2) then begin
                  if (((delta_values(cell_number-1, frame+1))-
(delta_values(cell_number-1, frame)) ge thresh1) OR $
                  (delta_values(cell_number-1, frame+2)-
delta_values(cell_number-1, frame) ge thresh2)) then begin
                        yes_no_values(cell_number-1, frame)=1
                  endif
            endif else begin  ;this takes care of a peak at point 1
                  if ((delta_values(cell_number-1, frame_no-1))-
(delta_values(cell_number-1, frame_no-2)) ge thresh1) then begin
                        yes_no_values(cell_number-1, frame_no-2)=1
                  endif
            endelse
      endfor
end

(thresh1 lt 0): begin                        ;for decreasing spikes, the threshold
will be less than 0
      for frame=1, frame_no-1 do begin       ;this doesn't consider a spike at the
first frame
            if (frame lt frame_no-2) then begin
                  if (((delta_values(cell_number-1, frame+1))-
(delta_values(cell_number-1, frame)) le thresh1) OR $
                        (delta_values(cell_number-1, frame+2)-
delta_values(cell_number-1, frame) le thresh2)) then begin
                                    yes_no_values(cell_number-1, frame)=1
                  endif
            endif else begin  ;this takes care of a peak at point 1
                  if ((delta_values(cell_number-1, frame_no-1))-
(delta_values(cell_number-1,frame_no-2)) le thresh1) then begin
                        yes_no_values(cell_number-1, frame_no-2)=1
                  endif
            endelse
```

```
      endfor
end
endcase
return, yes_no_values
end
```

```
;; many_cells_one_frame
;;
;; Creates a random distribution (via count_random_matches_many and
make_random_data)
;;  to compare the real data to- counts the number of times a minimum number of
;;  cells fire in one frame, or a set of frames given by the window size.
;;

function many_cells_one_frame, synchronous_cells, no_iterations, ge_or_eq_test,
window_size

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_create_dist, cells_active
common with_many, filename
common stockdata, symbol_array, date_array
;common with_correl_map_plan, cluster_array          ;uncomment this if you
want networks to have different linestyles in the correl_map_plane..

      ;; Initialize variables
      connections_array = bytarr(cell_no, cell_no)    ; if two cells are
significantly connected, this is 1; if not, 0
      yes_no_significance = yes_no_values             ; the new colored raster
plot is made from yes_no_significance.  This sets it to the original data
      cluster_array = intarr(cell_no, cell_no)        ; this holds the frame in
which the cells fire together
      true_hits = 0                                         ; number of networks
in the real data
      distances_array = [0.0]                         ; list of all network
connection distances
      areas_array = [0.0]                                   ; list of all
network areas
      size_array = intarr(frame_no)                         ; this counts only
those networks 'synchronous_cells'+ in size
      cluster_no = 0                                        ; this makes the
first network in solid lines

      ;; Creating the random distribution
      no_spikes = total(yes_no_values)                          ;this give the
total number of spikes by cells in the slice
      random_distribution = intarr(no_iterations)               ;this array
holds one random number of matches for every iteration
      for t = 0, no_iterations - 1 do begin
            make_random_data, seed, random_array, num_active_cells
            random_distribution(t) = count_random_matches_many(random_array,
synchronous_cells, ge_or_eq_test, window_size)
      endfor

      ;; Printing some of the data
      close,  1
```

```
      filename = pickfile(/write, file=(strcompress(string(synchronous_cells)) +
'_Stocks_Once'))
      if (filename eq '') then begin
            mess = WIDGET_MESSAGE('This data will not be saved',/INFORMATION)
            filename = 'Many_One.dat'
      endif
      openw, 1, filename            ;DO THIS-this is the temp file used for
writing into the xdisplay.  It can be saved to a further name.
      printf, 1, 'Statistical data for the Many/Once test.'
      printf, 1, 'One hit is when ', strcompress(synchronous_cells),' stocks
fire in one frame'
      printf, 1, 'Number of iterations:         ', no_iterations
      printf, 1, 'Window size: ', window_size
      printf, 1, 'Below is the list of stocks which spike in a given frame:'

      ;; Finding the real number of matches- first bins spikes according to
window size, no binning done when window is 0
      binned_yn_vals_array = intarr(cell_no, frame_no - (window_size * 2))
      for frame_counter = window_size, frame_no - (window_size + 1) do begin
            for window_counter = -window_size, window_size do begin
                  for cell_counter = 0, cell_no - 1 do begin
                        if (yes_no_values[cell_counter, frame_counter +
window_counter] ge 1) then begin
                              binned_yn_vals_array[cell_counter, frame_counter -
window_size] = 1
                        endif
                  endfor
            endfor
      endfor

      ;; Now looks for matches in the binned data- there may be false positives
for windows
      for frame_counter = 0, n_elements(binned_yn_vals_array[0,*]) - 1 do begin
            matches = total(binned_yn_vals_array(*, frame_counter))
            if (((ge_or_eq_test eq 0) and (matches ge synchronous_cells)) or
((ge_or_eq_test eq 1) and (matches eq synchronous_cells))) then begin
                  true_hits = true_hits + 1
                  size_array[frame_counter] = matches
                  printf, 1, ' _________'
                  if (window_size eq 0) then begin
                        printf, 1, strcompress(round(matches)), ' Stocks Spike
in Frame: ', strcompress(frame_counter)
                        correlated_cell_array = where(yes_no_values(*,
frame_counter) eq 1)
                  endif else begin
                        printf, 1, strcompress(round(matches)), ' Stocks Spike
at Approximately Frame: ', strcompress(frame_counter)
                        correlated_cell_array = where(binned_yn_vals_array(*,
frame_counter) eq 1)
                  endelse
                  network_xcors = intarr(n_elements(correlated_cell_array))
                  network_ycors = intarr(n_elements(correlated_cell_array))
                  for index = 0, matches - 1 do begin

                        printf, 1, '      Stock number: ',
correlated_cell_array(index) + 1
                        loc = correlated_cell_array(index)
```

```
                        for window_counter = frame_counter, frame_counter + (2 *
window_size) do begin
                              if (yes_no_values(correlated_cell_array(index),
window_counter) ge 1) then begin

      yes_no_significance(correlated_cell_array(index), window_counter) =
yes_no_significance(correlated_cell_array(index), window_counter) + 1
                              endif
                        endfor

                        network_xcors[index] = location(loc).coord[0]
                        network_ycors[index] = location(loc).coord[1]
                        printf, 1, '      X and Y: ', location(loc).coord
                        printf, 1, ' '
                        for index2 = index, matches - 1 do begin
                              if (index ne index2) then begin
                                    connection_distance =
sqrt(((double(location(correlated_cell_array[index]).coord[0]) -
double(location(correlated_cell_array[index2]).coord[0])) ^ 2) +
((double(location(correlated_cell_array[index]).coord[1]) -
double(location(correlated_cell_array[index2]).coord[1])) ^ 2))
                                    distances_array =
[distances_array,connection_distance]
                              endif
                              loc2 = correlated_cell_array(index2)
                              connections_array(loc, loc2) = 1
                              cluster_array(loc, loc2) = cluster_no
                        endfor
                  endfor

                  ;; find area of many cells once network
                  if ((n_elements(uniq(network_xcors, sort(network_xcors))) ge
3) and (n_elements(uniq(network_ycors, sort(network_ycors))) ge 3)) then begin
                        triangulate, network_xcors, network_ycors, triangles,
convex_hull ;; here we find the convex hull surrounding the network
                        convex_hull = [convex_hull,convex_hull[0]] ;; the last
vertex is the first for connections' sake
                        area = 0.0D
                        for i = 0, (n_elements(convex_hull) - 2) do begin
                              area = area +
((double(network_xcors[convex_hull[i]]) * double(network_ycors[convex_hull[i +
1]])) - (double(network_xcors[convex_hull[i + 1]]) *
double(network_ycors[convex_hull[i]])))
                        endfor
                        area = abs(area / 2)
                        areas_array = [areas_array,area]
                        printf, 1, '      Area of network (in pixels^2): ',
strcompress(area)
                        printf, 1, ' '
                  endif
                  cluster_no = cluster_no + 1
            endif
      endfor

      ;; draw raster plot before it checks to see if the random distribution has
a variance as the raster is independant of statistics
```

```
      draw_significance_raster, yes_no_significance

      ;; draw correlations map
      ;;window, /free, xsize=x_size, ysize=y_size, title='Correl Map of All
Many/Once Networks'     ;connections_array has the data of the cells connected
      ;;correl_map_image_plane, connections_array
      scorrel_map, symbol_array, connections_array

      ;; draw histogram of areas
      if (n_elements(areas_array) ge 2) then begin
            window, /free, title='Distribution of Many/Once Areas'
            hist_areas = histogram(areas_array[1:n_elements(areas_array) - 1],
binsize=5000, min=0, max=max(areas_array) + 5000)
            plot, lindgen(n_elements(hist_areas) + 2) * 5000L, hist_areas,
psym=10, title='Distribution of Many/Once Areas',$
                  yrange=[0, max(hist_areas) + 1], xrange=[0, max(areas_array) +
5000], ytitle='Number of Networks', xtitle='Area (Pixels ^ 2)'
      endif

      ;; check to see if the random distribution has any non zero values.  If
not, the moment cannot be defined for the distribution
      xmax=max(random_distribution)
      xmin=min(random_distribution)
      if (xmax eq xmin) then begin                              ;if the min and max
are the same, all of the elements are the same
            mess=WIDGET_MESSAGE('Random distribution has variance of zero.  Try
again with greater number of iterations', /error)
            printf, 1, 'ERROR!'
            printf, 1, 'Moment undefined for random distribution with variance
zero'
            close, 1
      endif else begin
            ;plot random distribution with a line for the actual value
            !p.multi = 0
            bin_size = 1
            hist = histogram(random_distribution, binsize = 1, min = 0, max = 2
* xmax + 1)       ;plot a histogram of the distribution
            window, /free, title='Many One Distribution'
            plot, hist, xtitle='number of hits', ytitle='frequency'
            y2 = total(random_distribution eq true_hits)
            if y2 le 0 then begin
      ;if there is no random data=true_hits, to draw the blue line
                  y2 = y2 + 1
            endif
            plots, [true_hits, true_hits], [0, y2], color=12            ;this
draws a line where the actual number of matches lies

            ;calculating the number of spikes per cell per second
            spikes_per_cell_per_second = (no_spikes / cells_active) / (frame_no
* time_resolution)

            ;calculating the p value, standard dev. for the data
            stats = moment(random_distribution, sdev=sdev)
            no_points_right = total(random_distribution ge true_hits)
            p_value = no_points_right / no_iterations

            ;writing more data to the file:
```

```
            printf, 1, 'Total number of active stocks:            ',
strcompress(cells_active)
            printf, 1, 'Total number of frames:             ',
strcompress(frame_no)
            printf, 1, 'Total number of spikes:             ',
strcompress(no_spikes)
            printf, 1, 'Mean expected matches:              ',
strcompress(stats(0))
            printf, 1, 'Variance:                                ',
strcompress(stats(1))
            printf, 1, 'Standard deviation:                      ',
strcompress(sdev)
            printf, 1, 'Actual matches (no. of networks):   ',
strcompress(true_hits)
            printf, 1, 'Number of networks/number of stocks:     ',
strcompress(true_hits / cells_active)
            printf, 1, 'Normalized number of networks:           ',
strcompress(true_hits / cells_active / frame_no)
            printf, 1, 'Actual/expected:                    ',
strcompress(true_hits / stats(0))
            printf, 1, 'Standard error for ratio:                ',
strcompress(sdev / stats(0))
            if (n_elements(where(size_array)) ge 2) then begin
                  network_stats = moment(size_array(where(size_array)),
sdev=sdev_size_array)
                  printf, 1, 'Mean stocks in a network:                    ',
strcompress(network_stats[0])
                  printf, 1, 'Standard deviation:                      ',
strcompress(sdev_size_array)
            endif
            printf, 1, ' '
            printf, 1, 'Spike firing rate:                       ',
strcompress(spikes_per_cell_per_second)
            if (n_elements(distances_array) gt 2) then begin
                  distance_stats =
moment(distances_array[1:n_elements(distances_array) - 1],
sdev=sdev_distances_array)
                  printf, 1, 'Mean connection distance:                    ',
strcompress(distance_stats[0])
                  printf, 1, 'Standard deviation:                      ',
strcompress(sdev_distances_array)
            endif else begin
                  if (n_elements(distances_array) gt 1) then begin
                        printf, 1, 'Connection distance:                 ',
strcompress(distances_array[1])
                  endif
            endelse
            if (n_elements(areas_array) gt 2) then begin
                  area_stats = moment(areas_array[1:n_elements(areas_array) -
1], sdev=sdev_areas_array)
                  printf, 1, 'Mean area:                               ',
strcompress(area_stats[0])
                  printf, 1, 'Standard deviation:                          ',
strcompress(sdev_areas_array)
            endif
            printf, 1, 'Significance p-value:               ',
strcompress(p_value)
```

```
            close, 1
      endelse

      return, [random_array]
end
;-------------------------------------------------------------------
;     Name: many_one_widget
;     Description: Through this widget, we can find how significant many cells
firing in one frame is.  The user
;           can choose how many cells consititutes a hit and how many iterations
the program should run through
;           to create the random_distribution.
;

pro many_one_widget

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size

      many_one_base=WIDGET_BASE(/COLUMN, title='Significance of many stocks
spiking together in one frame')
      many_one_field1=CW_FIELD(many_one_base, /RETURN_EVENTS, /INTEGER,
TITLE='Number of stocks to spike in one frame to count as a hit:',$
                        VALUE=4, UVALUE=0)
      ge_or_eq_bgroup=CW_BGROUP(many_one_base, ['This Many or More','Exactly
This Many'], /ROW, /EXCLUSIVE, SET_VALUE=0)
      many_one_field2=CW_FIELD(many_one_base, /RETURN_EVENTS, /INTEGER,
TITLE='Number of iterations:',$
                        VALUE=1000, UVALUE=0)
      window_size_field=CW_FIELD(many_one_base, /RETURN_EVENTS, /INTEGER,
TITLE='Window size for hits:', VALUE=0, UVALUE=0)
      many_one_button1=WIDGET_BUTTON(many_one_base, VALUE='Find significance',
UVALUE=2)
      WIDGET_CONTROL, /realize, many_one_base
      many_one_state={many_one_field1:many_one_field1,$
                        many_one_field2:many_one_field2,$
                        window_size_field:window_size_field,$
                        many_one_button:many_one_button1,$
                        ge_or_eq_bgroup:ge_or_eq_bgroup}
      WIDGET_CONTROL, WIDGET_INFO(many_one_base, /CHILD),
SET_UVALUE=many_one_state
      xmanager, 'many_one_widget', many_one_base
end

;-------------------------------------------------------------------
; NAME: many_one_widget_event, event

pro many_one_widget_event, event
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_many, filename

      stateholder3=WIDGET_INFO(event.handler, /CHILD)
      WIDGET_CONTROL, stateholder3, GET_UVALUE=many_one_state
      WIDGET_CONTROL, many_one_state.many_one_field1,
GET_VALUE=synchronous_cells
```

```
      WIDGET_CONTROL, many_one_state.many_one_field2, GET_VALUE=no_of_iterations
      WIDGET_CONTROL, many_one_state.window_size_field, GET_VALUE=window_size
      WIDGET_CONTROL, many_one_state.ge_or_eq_bgroup, GET_VALUE=ge_or_eq_test

      if (event.id eq many_one_state.many_one_button) then begin
            if ((synchronous_cells le 0) OR (no_of_iterations le 1)) then begin
                  mess=WIDGET_MESSAGE('Invalid fields specified!', /error)
            endif else begin
                  random_array = many_cells_one_frame(synchronous_cells,
no_of_iterations, ge_or_eq_test, window_size)
                  WIDGET_CONTROL, event.top, /hourglass
                  close, 1
                  openr, 1, filename
                  stat_stuff = fstat(1)
                  file_size = stat_stuff.size
                  close, 1
                  if (file_size gt 8112) then begin
                        mess = WIDGET_MESSAGE('File is too large to display
through a widget. Open it manually. If you did not save it, it is named
Many_One.dat', /INFORMATION)
                  endif else begin
                        xdisplayfile, filename, title = "Statistical Data for
Many/One", group = event.top, width = 75, height = 50
                  endelse
            endelse
      endif
end
```

```
pro MultiStock

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common old_skool_data, original_data
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common test, str_ing, state3
common choose_cell_convert_delta, filepath1
common textfile_vars, text_flag, diode_array, max_num_of_diodes


      ;; Init all the common block vars above! (except state3)
      pixel_array = [0]
      yes_no_values = [0]
      original_data = [0]
      coef = [0]
      location = 0
      cell_no = 1
      frame_no = 1
      file_name = ''
      total_frame_no = 300
      time_resolution = 1.0
      x_size = 0
      y_size = 0
      box_size = 0
      cells_defined = 0
```

```
spikes_defined = 0
single_plot_defined = 0
correl_coef_defined = 0
rms_threshold = [0]
threshold1 = [0]
threshold2 = [0]
str_ing = 'Movie.'
buqing_flag= 0
diode_array = [0]
filepath1 = 'dev/null'  ;; but where else to send our data?
text_flag = 0
diode_array = [0]


base=WIDGET_BASE(/column, title='MultiStock')
;; setting uvalue=0 below tells the event handler that we're going to
refer to the buttons by name,
;; i.e. pass the button name as a string as the event.
b=cw_bgroup(base, /row, [     'New Experiment',$
                              'Save as XDR',$
                              'Load from XDR',$
                              'Exit'], /return_name, UVALUE=0)
;this is the structure that has the information for the pulldown menu
stuff= {cw_pdmenu_s, flags:0, name:''}
details=[   { cw_pdmenu_s, 1, 'Find Spikes'}, $
            { cw_pdmenu_s, 0, 'Root Mean Squared'},$
            { cw_pdmenu_s, 2, 'Intensity Threshold'},$
            { cw_pdmenu_s, 0, 'Delete Spikes'}, $
            { cw_pdmenu_s, 1, 'Plotting' }, $
            { cw_pdmenu_s, 0, 'Single Plots'},$
            { cw_pdmenu_s, 0, 'All Plots'},$
            { cw_pdmenu_s, 0, 'All Plots-3D'},$
            { cw_pdmenu_s, 0, 'Superimposed Plots'},$
            { cw_pdmenu_s, 0, 'Raster Plot'},$
            { cw_pdmenu_s, 2, 'Overall Behavior'}, $
            { cw_pdmenu_s, 0, 'Load from Text File'}]

pull_down=cw_pdmenu(base, details, /return_full_name, UVALUE=12)

stuff2= {cw_pdmenu_s2, flags:0, name:''}
details=[   { cw_pdmenu_s2, 1, 'Test Significance'}, $
            { cw_pdmenu_s2, 0, 'General Significance'}, $
            { cw_pdmenu_s2, 0, 'Many Stocks One Time'}, $
            { cw_pdmenu_s2, 2, 'Two Stocks Many Times'}, $
            { cw_pdmenu_s2, 0, 'Build Correlation Map'},$
            { cw_pdmenu_s2, 0, 'Cross Correlogram'}]

pull_down2=cw_pdmenu(base, details, /return_full_name, UVALUE=13)

stuff3= {cw_pdmenu_s3, flags:0, name:''}
details=[   { cw_pdmenu_s3, 0, 'Color Tables'}, $
            { cw_pdmenu_s3, 0, 'Return to IDL'},$
            { cw_pdmenu_s3, 0, 'Load from Excel'}]

pull_down3=cw_pdmenu(base, details, /return_full_name, UVALUE=14)
```

```
      namesid=cw_field(base, title='Filename Root                          ',
UVALUE=2, VALUE=str_ing, /STRING, /return_events)
      frameid=CW_FIELD(base, title='Total Number of Days
',VALUE=total_frame_no, /INTEGER, /RETURN_EVENTS)
      timeid=CW_FIELD(base, title='Time Resolution
',VALUE=time_resolution, /FLOATING, /RETURN_EVENTS)

      widget_control, /realize, base  ; make the widget visible

      state={name:namesid, frame:frameid, time:timeid}
      WIDGET_CONTROL, WIDGET_INFO(base, /Child), SET_UVALUE=state

      xmanager, 'MultiStock', base        ;register this widget with the
xmanager so it can call the event
end
```

```
pro MultiStock_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common old_skool_data, original_data
common test, str_ing, state3
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common choose_cell_convert_delta, filepath1
common markov2, yes_no_values3, yes_no_temp,cell_no_temp,frame_no_temp
common textfile_vars, text_flag, diode_array, max_num_of_diodes

      stateholder3=WIDGET_INFO(event.handler, /Child)
      WIDGET_CONTROL, stateholder3, GET_UVALUE=state3       ;find what is in
/Child
      WIDGET_CONTROL, state3.name, GET_VALUE=file_name
      WIDGET_CONTROL, state3.frame, GET_VALUE=total_frame_no
      WIDGET_CONTROL, state3.time, GET_VALUE=time_resolution


      if ((size(event.value))[1] eq 7) then begin  ;; i.e. if we've pressed a
button which returns a string as its value...
            case event.value of

                  ;; --------FIRST ROW ITEMS---------

                  'New Experiment': begin

                        end

                  'Save as XDR': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              data_file=pickfile(/write,
file='Variables_from_exp_#_', title='Create the saved variables file')
                              if (data_file eq '') then begin
```

```
                                    mess=WIDGET_MESSAGE('No saved variables file
specified.', /INFORMATION)
                              endif else begin
                                    SAVE, /VARIABLES, FILENAME=data_file, all,
/verbose
                                    mess=WIDGET_MESSAGE('Data from this
experiment saved!', /INFORMATION)
                              endelse
                        endelse
                  end

                  'Load from XDR': begin
                        state4=state3
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              data_file=pickfile(/read, title='Select the saved
variables file', GET_PATH=filepath1)
                              if (data_file eq '') then begin
                                    mess=WIDGET_MESSAGE('No saved variables file
specified.', /INFORMATION)
                              endif else begin
                                    rms_threshold = [0] ;; reset it so that it
can be properly reinitialized if need be
                                    restore, data_file
                                    size_of_loc=size(location)
                                    temp_cell=size_of_loc(1)
                                    size_of_y_n_v=size(yes_no_values)
                                    frame_no=size_of_y_n_v(2)
                                    cell_no=size_of_y_n_v(1)
                                    if (temp_cell ne cell_no) then begin
                                          mess=WIDGET_MESSAGE('Something is
wrong with your saved variables file!', /Error)
                                    endif else begin
                                          ;; cells_defined = 1
                                          ;; spikes_defines = 1
                                          single_plot_defined=0
                                          text_flag = 0
                                          if (n_elements(threshold1) eq 1) then
begin ;; back-compatibility check
                                                temp1 = threshold1
                                                temp2 = threshold2
                                                threshold1 = fltarr(cell_no)
                                                threshold2 = fltarr(cell_no)
                                                for cell = 0, cell_no - 1 do
begin
                                                      threshold1[cell] = temp1
                                                      threshold2[cell] = temp2
                                                endfor
                                          endif
                                          ;; more back-compatibility...
                                          if (n_elements(rms_threshold) eq 1)
then begin
                                                rms_threshold = fltarr(cell_no)
```

```
                                                    for cell = 0, cell_no - 1 do
begin
                                                          rms_threshold[cell] = 2.0
                                                    endfor
                                              endif
                                              WIDGET_CONTROL, state4.name,
SET_VALUE=str_ing
                                              WIDGET_CONTROL, state4.frame,
SET_VALUE=frame_no
                                              WIDGET_CONTROL, state4.time,
SET_VALUE=time_resolution
                                              mess=WIDGET_MESSAGE('Data from
previous experiment loaded!', /INFORMATION)
                                        endelse
                                  endelse
                                  data_file=0
                            endelse
                      end

                      'Exit':begin
                            WIDGET_CONTROL, /DESTROY, event.top
                      end

                      ;; --------SECOND ROW ITEMS---------

               'Find Spikes.Root Mean Squared': begin
                            if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                                  mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                            endif else begin
                                  if (cells_defined ne 1) then begin
                                        mess=WIDGET_MESSAGE('You have to find cells
before you find spikes!', /error)
                            endif else begin
                                        rms_spikes_widget
                                  endelse
                            endelse
                      end

                      'Find Spikes.Intensity Threshold': begin
                            if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                                  mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                            endif else begin
                                  if (cells_defined ne 1) then begin
                                        mess=WIDGET_MESSAGE('You have to find cells
before you find spikes!', /error)
                            endif else begin
                                        choose_threshold
                                  endelse
                            endelse
                      end

                    'Delete Spikes': begin
                            if (spikes_defined ne 1) then begin
```

```
                              mess = WIDGET_MESSAGE('You have to find spikes
before you can delete any!', /error)
                        endif else begin
                              delete_spikes_widget
                        endelse
                  end

                  'Plotting.Single Plots': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (spikes_defined ne 1) then begin
                              mess=WIDGET_MESSAGE('You have to find spikes
before you can plot cells!', /Error)
                              endif else begin
                                    pixel_vs_time_widget
                              endelse
                        endelse
                  end

                  'Plotting.All Plots': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                        if (single_plot_defined ne 1) then begin
                                    mess=WIDGET_MESSAGE('You have to plot single
cells before you can plot all cells!', /Error)
                        endif else begin
                              draw_spikes_many_pages, pixel_array,
yes_no_values, cell_no, frame_no, time_resolution
                              endelse
                        endelse
                  end

                  'Plotting.All Plots-3D': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                        if (cells_defined ne 1) then begin
                                    mess=WIDGET_MESSAGE('You have to plot single
cells before you can plot all cells!', /Error)
                        endif else begin
                              draw_3D_plot, pixel_array
                              endelse
                        endelse
                  end

                  'Plotting.Superimposed Plots': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
```

```
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (cells_defined ne 1) then begin
                                    mess=WIDGET_MESSAGE('You must find cells
before you can plot them!', /Error)
                              endif else begin
                                    draw_spikes_many_cell_delta, pixel_array,
cell_no, frame_no, time_resolution
                              endelse
                        endelse
                  end

                  'Plotting.Raster Plot': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                        if (spikes_defined ne 1) then begin
                              mess=WIDGET_MESSAGE('You have to find spikes
before drawing a raster plot!', /Error)
                              endif else begin
                                    if (total(yes_no_values) eq 0) then begin
                                          mess=WIDGET_MESSAGE('None of the cells
have any spikes! Try lowering the thresholds', /INFORMATION)
                                    endif else begin
                                    draw_raster
                                    endelse
                              endelse
                        endelse
                  end

                  'Plotting.Overall Behavior': begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (single_plot_defined ne 1) then begin
                                    mess=WIDGET_MESSAGE('You have to plot single
cells before you can plot all cells!', /Error)
                        endif else begin
                                    summed_spikes
                              endelse
                  endelse
                  end

                  'Load from Text File': begin
                        end

                  ;; --------THIRD ROW ITEMS----------

                  'Test Significance.General Significance':begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
```

```
                              mess = WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (spikes_defined ne 1) then begin
                                    mess = WIDGET_MESSAGE('You have to find
spikes before calculating the significance of correlations!', /error)
                              endif else begin
                                    gen_sig_widget
                              endelse
                        endelse
                  end

                  'Test Significance.Many Stocks One Time':begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess = WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (spikes_defined ne 1) then begin
                                    mess = WIDGET_MESSAGE('You have to find
spikes before calculating the significance of correlations!', /error)
                              endif else begin
                                    many_one_widget
                              endelse
                        endelse
                  end

                  'Test Significance.Two Stocks Many Times':begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess = WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (spikes_defined ne 1) then begin
                                    mess = WIDGET_MESSAGE('You have to find
spikes before calculating the significance of correlations!', /error)
                              endif else begin
                                    two_many_widget
                              endelse
                        endelse
                  end

                  'Build Correlation Map':begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess = WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (spikes_defined ne 1) then begin
                                    mess = WIDGET_MESSAGE('You have to find
spikes before calculating correlation coefficients!', /error)
                              endif else begin
                                    widget_analyze
                              endelse
                        endelse
                  end
```

```
                  'Cross Correlogram':begin
                        if (total_frame_no eq 0) or (time_resolution eq 0) then
begin
                              mess=WIDGET_MESSAGE('Enter frame number and time
resolution before proceeding!', /ERROR)
                        endif else begin
                              if (spikes_defined ne 1) then begin
                                    mess=WIDGET_MESSAGE('You have to find spikes
before you can draw correlograms!', /Error)
                              endif else begin
                                    yes_no_values3 = yes_no_values
                                    choose_crosscorr
                              endelse
                        endelse
                  end

                  ;; --------FOURTH ROW ITEMS---------

                  'Color Tables':begin
                        xloadct
                  end

                  'Return to IDL':begin
                        retall
                  end

                  'Load from Excel':begin
                        load_and_convert_excelfile
                  end

                  else:

            endcase
      endif
end
```

```
; Object: Change pixel_vs time_widget so that you can change the threshold for
certain cells
; while not doing so for the whole array of cell.
; This will work by having the threshold widget work as it did before, and then
also
; creating an option to rethreshold particular cells if necessary.
; Through this method, the spikes will be recalculated every time the graph is
displayed.
; Or it can have a statement which checks whether the threshold chosen for that
cell is the
; universal one, and in that case it will not rethreshold.
;_____________WIDGET FOR DRAWING A SINGLE
SPIKE_____________________________________

pro pixel_vs_time_widget
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_choose_cells_com, threshold1, threshold2, rms_threshold

      deep_base=WIDGET_BASE(/row, title='Plot of a Single Stock', /scroll)
```

```
      left_base=WIDGET_BASE(deep_base, /column)
      right_base=WIDGET_BASE(deep_base, /column)
      draw=WIDGET_DRAW(left_base, xsize=450, ysize=450)
      ;; kludge to fix the 'only one cell selected' bug
      if (cell_no gt 1) then begin
            slide=WIDGET_SLIDER(left_base, value=1, maximum=cell_no, minimum=1,$
                                    title='Choose the stock you want to plot',
uvalue='slider_pressed')
      endif else begin
            slide=WIDGET_TEXT(left_base, value='Only one stock selected...')
      endelse
      spike_max=CW_FIELD(right_base, title='Y Max for Spike:', VALUE=0,
/RETURN_EVENTS, /FLOATING)
      spike_min=CW_FIELD(right_base, title='Y Min for Spike: ', VALUE=-20,
/RETURN_EVENTS, /FLOATING)
      ymax=CW_FIELD(right_base, title='Y Range Max:',
VALUE=max([transpose(pixel_array(0, *)), 10]), /RETURN_EVENTS, /FLOATING)
      ymin=CW_FIELD(right_base, title='Y Range Min: ',
VALUE=min([transpose(pixel_array(0, *)), -10]), /RETURN_EVENTS, /FLOATING)
      xmax=CW_FIELD(right_base, title='X Range Max:',
VALUE=total_frame_no*time_resolution, /RETURN_EVENTS, /FLOATING)
      xmin=CW_FIELD(right_base, title='X Range Min: ', VALUE=0, /RETURN_EVENTS,
/FLOATING)
      median_filter_window=CW_FIELD(right_base, title='Window for Median Filter:
', VALUE=20, /RETURN_EVENTS, /FLOATING)
      smoothing_window=CW_FIELD(right_base, title='Window for Mean Smoothing:',
VALUE=3, /RETURN_EVENTS, /FLOATING)
      buttons1=cw_bgroup(right_base, /row, ['Smooth', 'Median Filter','Restore
Original Waveform'], $
            BUTTON_UVALUE=['smooth_pressed', 'filter_pressed',
'restore_pressed'], UVALUE=5)
      rms_threshold_box=CW_FIELD(right_base, title='RMS Threshold:', VALUE=2.0,
$
            UVALUE=19, /RETURN_EVENTS, /FLOATING)
      rms_spikes_type_bgroup=CW_BGROUP(right_base, ['Positive Spikes','Negative
Spikes'], /ROW, /EXCLUSIVE, SET_VALUE=0)
      threshold_one=CW_FIELD(right_base, title='2 Frame Intensity Diff
Threshold:', VALUE=threshold1[0], $
            UVALUE=6, /RETURN_EVENTS, /FLOATING)
      threshold_two=CW_FIELD(right_base, title='3 Frame Intensity Diff
Threshold:', VALUE=threshold2[0], $
            UVALUE=7, /RETURN_EVENTS, /FLOATING)
      buttons2=cw_bgroup(right_base, /row, ['Plot with RMS', 'Plot with
Intensity Difference'], $
            BUTTON_UVALUE=['RMS_plot_pressed', 'diff_plot_pressed'], UVALUE=3)
      delete_spike=CW_FIELD(right_base,title='Delete Spike Number:', VALUE='0',
UVALUE=2,$
            /RETURN_EVENTS, /INTEGER)
      buttons3=cw_bgroup(right_base, /row, ['Delete Stock', 'Delete All Trailer
Spikes'], $
            BUTTON_UVALUE=['delete_cell_pressed',
'delete_trailer_spikes_pressed'], UVALUE=4)
      holder={draw:draw, slide:slide, delete_spike:delete_spike,$
               spike_max:spike_max, spike_min:spike_min,$
               ymax:ymax, ymin:ymin, xmin:xmin, xmax:xmax,$
               rms_spikes_type_bgroup:rms_spikes_type_bgroup, $
```

```
              median_filter_window:median_filter_window,
smoothing_window:smoothing_window, $
              rms_threshold_box:rms_threshold_box, threshold_one:threshold_one,
threshold_two:threshold_two}
      WIDGET_CONTROL, deep_base, /realize
      WIDGET_CONTROL, deep_base, set_uvalue=holder

      plot, findgen(frame_no) * time_resolution, pixel_array(0, *),
yrange=[min([transpose(pixel_array(0, *)), -10]), max([transpose(pixel_array(0,
*)), 10])],$
            xrange=[0, total_frame_no * time_resolution], title=1, ytitle='%
Change in Index', xtitle='Time (days)', FONT=-1
      for frames=1, frame_no - 1 do begin                    ;this doesn't
consider a spike at the 1st frame
            if (yes_no_values(0, frames) ne 0) then begin
                  plots, [frames * time_resolution, frames * time_resolution],
[0, -20]
            endif
      endfor

      XMANAGER, 'pixel_vs_time_widget', deep_base
  end
```

```
pro pixel_vs_time_widget_event, event
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common share_w_pixel_vs_time_widget_com, y_min, y_max, spike_min, spike_max,
x_max, x_min
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common old_skool_data, original_data
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined

      WIDGET_CONTROL, event.top, get_uvalue=state
      WIDGET_CONTROL, event.id, GET_UVALUE=uval
      WIDGET_CONTROL, state.draw, GET_VALUE=window
      WIDGET_CONTROL, state.slide, GET_VALUE=number_1

      if (string(uval) eq 'slider_pressed') then begin
            WIDGET_CONTROL, state.delete_spike, set_value = 0
            spike_1 = 0
            WIDGET_CONTROL, state.ymax, set_value = max(pixel_array((number_1 -
1), *))
            y_max = max(pixel_array((number_1 - 1), *))
            WIDGET_CONTROL, state.ymin, set_value = min(pixel_array((number_1 -
1), *))
            y_min = min(pixel_array((number_1 - 1), *))

            ;; check for 10,-10 boundary... cheap...
            WIDGET_CONTROL, state.ymax, set_value = max([y_max, 10])
            y_max = max([y_max, 10])
            WIDGET_CONTROL, state.ymin, set_value = min([y_min, -10])
            y_min = min([y_min, -10])
```

```
            WIDGET_CONTROL, state.rms_threshold_box,
SET_VALUE=rms_threshold[number_1 - 1]
            rms_t = rms_threshold[number_1 - 1]
            WIDGET_CONTROL, state.threshold_one, SET_VALUE=threshold1[number_1 -
1]
            t1 = threshold1[number_1 - 1]
            WIDGET_CONTROL, state.threshold_two, SET_VALUE=threshold2[number_1 -
1]
            t2 = threshold2[number_1 - 1]
      endif

      WIDGET_CONTROL, state.delete_spike, GET_VALUE=spike_1
      WIDGET_CONTROL, state.median_filter_window, GET_VALUE=median_filter_window
      WIDGET_CONTROL, state.smoothing_window, GET_VALUE=smoothing_window
      WIDGET_CONTROL, state.rms_threshold_box, GET_VALUE=rms_t
      WIDGET_CONTROL, state.threshold_one, GET_VALUE=t1
      WIDGET_CONTROL, state.threshold_two, GET_VALUE=t2
      WIDGET_CONTROL, state.ymin, GET_VALUE=y_min
      WIDGET_CONTROL, state.ymax, GET_VALUE=y_max
      WIDGET_CONTROL, state.xmin, GET_VALUE=x_min
      WIDGET_CONTROL, state.xmax, GET_VALUE=x_max
      WIDGET_CONTROL, state.spike_min, GET_VALUE=spike_min
      WIDGET_CONTROL, state.spike_max, GET_VALUE=spike_max

      wset, window                                        ;even if you open other
windows, this will still plot in the original window

      ;; trailing spike clumps deletion
      if (string(event.value) eq 'delete_trailer_spikes_pressed') then begin
            for frame_counter = (frame_no - 1), 2, -1 do begin
                  if ((yes_no_values[number_1 - 1, frame_counter] eq 1) and
((yes_no_values[number_1 - 1, frame_counter - 1] eq 1) or
(yes_no_values[number_1 - 1, frame_counter - 2] eq 1))) then begin
                        yes_no_values[number_1 - 1, frame_counter] = 0
                  endif
            endfor
            draw_spikes, number_1
      endif

      ;; if we're re-plotting, update the yes_no_values to current threshold
      if (string(event.value) eq 'diff_plot_pressed') then begin
            WIDGET_CONTROL, state.delete_spike, set_value = 0
            spike_1 = 0
            if ((t1 eq 0) OR (t2 eq 0)) then begin
                  mess=WIDGET_MESSAGE('Threshold cannot be 0!', /Error)
                  WIDGET_CONTROL, state.threshold_one,
SET_VALUE=threshold1[number_1 - 1]
                  WIDGET_CONTROL, state.threshold_two,
SET_VALUE=threshold2[number_1 - 1]
                  t1 = threshold1[number_1 - 1]
                  t2 = threshold2[number_1 - 1]
            endif else begin
                  threshold1[number_1 - 1] = t1
                  threshold2[number_1 - 1] = t2
                  yes_no_values=make_single_binary(pixel_array, yes_no_values,
number_1, frame_no, threshold1[number_1 - 1], threshold2[number_1 - 1])
            endelse
```

```
      endif

      if (string(event.value) eq 'RMS_plot_pressed') then begin
            WIDGET_CONTROL, state.delete_spike, set_value = 0
            spike_1 = 0
            WIDGET_CONTROL, state.rms_spikes_type_bgroup, GET_VALUE=pos_or_neg
            rms_threshold[number_1 - 1] = rms_t ;; update new threshold with
value in textbox

            if (pos_or_neg eq 0) then begin
                  cell_stats = moment(pixel_array[number_1 - 1,*],
sdev=cell_dff_stddev)
                  cell_dff_mean = cell_stats[0]
                  for frame_counter = 1, frame_no - 1 do begin
                        if (pixel_array[number_1 - 1,frame_counter] gt
(cell_dff_mean + (rms_threshold[number_1 - 1] * cell_dff_stddev))) then begin
                              yes_no_values[number_1 - 1, frame_counter] = 1
                        endif else begin
                              yes_no_values[number_1 - 1, frame_counter] = 0
                        endelse
                  endfor
            ;; negative spikes is pos_or_neg = 1
            endif else begin
                  cell_stats = moment(pixel_array[number_1 - 1,*],
sdev=cell_dff_stddev)
                  cell_dff_mean = cell_stats[0]
                  for frame_counter = 1, frame_no - 1 do begin
                        if (pixel_array[number_1 - 1,frame_counter] lt
(cell_dff_mean - (rms_threshold[number_1 - 1] * cell_dff_stddev))) then begin
                              yes_no_values[number_1 - 1, frame_counter] = 1
                        endif else begin
                              yes_no_values[number_1 - 1, frame_counter] = 0
                        endelse
                  endfor
            endelse
      endif

      ;; filtering- subtractive median filter: see filter_median_subtractive.pro
      if (string(event.value) eq 'filter_pressed') then begin
            if ((median_filter_window lt 2) or (median_filter_window ge
frame_no)) then begin ;;error checking
                  error_message = WIDGET_MESSAGE('Window size for filtering must
be greater than 1 and less than the number of frames!', /INFORMATION)
            endif else begin
                  filter_median_subtractive, pixel_array[number_1 - 1,*],
median_filter_window, filtered_cell
                  pixel_array[number_1 - 1,*] = filtered_cell
            endelse
      endif

      ;; smoothing, via the IDL function 'smooth'
      if (string(event.value) eq 'smooth_pressed') then begin
            if ((smoothing_window lt 2) or (smoothing_window ge frame_no)) then
begin ;;error checking
                  error_message = WIDGET_MESSAGE('Window size for smoothing must
be greater than 1 and less than the number of frames!', /INFORMATION)
            endif else begin
```

```
                  pixel_array[number_1 - 1,*] = smooth(pixel_array[number_1 -
1,*], smoothing_window)
            endelse
      endif

      ;; back to basics
      if (string(event.value) eq 'restore_pressed') then begin
            pixel_array[number_1 - 1, *] = original_data[number_1 - 1, *]

            WIDGET_CONTROL, state.ymax, set_value = max(pixel_array((number_1 -
1), *))
            y_max = max(pixel_array((number_1 - 1), *))
            WIDGET_CONTROL, state.ymin, set_value = min(pixel_array((number_1 -
1), *))
            y_min = min(pixel_array((number_1 - 1), *))

            ;; check for 10,-10 boundary... cheap...
            WIDGET_CONTROL, state.ymax, set_value = max([y_max, 10])
            y_max = max([y_max, 10])
            WIDGET_CONTROL, state.ymin, set_value = min([y_min, -10])
            y_min = min([y_min, -10])
      endif

      ;; correct for zero-threshold anyways, it's cheap
      if ((t1 eq 0) OR (t2 eq 0)) then begin
            mess=WIDGET_MESSAGE('Threshold cannot be 0!', /Error)
            WIDGET_CONTROL, state.threshold_one, SET_VALUE=threshold1[number_1 -
1]
            WIDGET_CONTROL, state.threshold_two, SET_VALUE=threshold2[number_1 -
1]
      endif

      ;; update the list of spikes
      time_array = where(yes_no_values(number_1 - 1, *) eq 1)     ;this finds
the location of the spikes for each cell
      number_spikes = n_elements(time_array)

      ;; check for spike deletion
      if ((time_array(0) ne -1) AND (spike_1 ne 0)) then begin
            if ((spike_1 gt number_spikes) OR (spike_1 lt 1)) then begin
                  mess=WIDGET_MESSAGE('You entered an invalid spike number.')
                  WIDGET_CONTROL, state.delete_spike, set_value = 0
                  spike_1 = 0
            endif else begin
                  yes_no_values(number_1 - 1, time_array(spike_1 - 1)) = 0
                  draw_spikes, number_1                              ;that cell
will now be displayed
                  WIDGET_CONTROL, state.delete_spike, set_value = 0
                  spike_1 = 0
            endelse
      ;; here is where all the plotting gets done
      endif else begin                                    ;if no spikes are to be
deleted
            draw_spikes, number_1
            WIDGET_CONTROL, state.delete_spike, set_value = 0
            spike_1 = 0
      endelse
```

```
      ;; check for cell deletion
      if (string(event.value) eq 'delete_cell_pressed') then begin
            yes_no_values(number_1 - 1,*) = 0
            draw_spikes, number_1
            mess=WIDGET_MESSAGE(string('Stock', byte(number_1),' has been
successfully deleted.'), /information)
      endif

      single_plot_defined=1                        ;this is sent to Multicell_event
so the other programs now that single plots has been done
 end
```

```
; Name: rms_spikes_widget
;
; Description: a new, improved method of spike detection! Finds the mean and a
number of standard deviations
;     (generally 2... the Root Mean Square version of signal/noise handling),
and any points beyond this
;     threshold are considered spikes.
;

pro rms_spikes_widget
common rms_spikes, rms_spikes_state

      rms_spikes_base=WIDGET_BASE(/COLUMN, title='Find Spikes- RMS')
      rms_spikes_button=WIDGET_BUTTON(rms_spikes_base, VALUE='Find Spikes',
UVALUE=2)
      rms_spikes_type_bgroup=CW_BGROUP(rms_spikes_base, ['Positive
Spikes','Negative Spikes'], /ROW, /EXCLUSIVE, SET_VALUE=0)
      rms_spikes_threshold_field=CW_FIELD(rms_spikes_base, /RETURN_EVENTS,
 /FLOATING, TITLE='Threshold (Number of Std Devs)',$
            VALUE=2.0, UVALUE=0)
      rms_spikes_median_filter_window_size_field=CW_FIELD(rms_spikes_base,
/RETURN_EVENTS, /FLOATING, TITLE='Window Size for Median Filter',$
            VALUE=20.0, UVALUE=0)
      rms_spikes_text=WIDGET_TEXT(rms_spikes_base, VALUE='Set window size to 0
to prevent filtering...')

      WIDGET_CONTROL, /realize, rms_spikes_base
      rms_spikes_state={      rms_spikes_base:rms_spikes_base, $
                              rms_spikes_button:rms_spikes_button, $
                              rms_spikes_type_bgroup:rms_spikes_type_bgroup, $

      rms_spikes_threshold_field:rms_spikes_threshold_field, $

      rms_spikes_median_filter_window_size_field:rms_spikes_median_filter_window
_size_field}
      WIDGET_CONTROL, WIDGET_INFO(rms_spikes_base, /CHILD),
SET_UVALUE=rms_spikes_state

      xmanager, 'rms_spikes_widget', rms_spikes_base
 end
```

```
; Name: rms_spikes_widget_event

pro rms_spikes_widget_event, event
```

```
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common rms_spikes, rms_spikes_state

      rms_state_stuff=WIDGET_INFO(event.handler, /Child)
      WIDGET_CONTROL, rms_state_stuff, GET_UVALUE=rms_spikes_state      ;find
what is in /Child

      if (event.id eq rms_spikes_state.rms_spikes_button) then begin
            ;; read interface data...
            WIDGET_CONTROL, rms_spikes_state.rms_spikes_type_bgroup,
GET_VALUE=pos_or_neg
            WIDGET_CONTROL, rms_spikes_state.rms_spikes_threshold_field,
GET_VALUE=threshold
            WIDGET_CONTROL,
rms_spikes_state.rms_spikes_median_filter_window_size_field,
GET_VALUE=window_size

            for i = 0, cell_no - 1 do begin
                  rms_threshold[i] = threshold
            endfor

            if (window_size gt 0) then begin
                  filter_median_subtractive, pixel_array, window_size,
pixel_array
            end

            yes_no_values=intarr(cell_no,frame_no)

            ;; positive spikes is pos_or_neg = 0
            if (pos_or_neg eq 0) then begin
                  for cell_counter = 0, cell_no - 1 do begin
                        cell_stats = moment(pixel_array[cell_counter,*],
sdev=cell_dff_stddev)
                        cell_dff_mean = cell_stats[0]
                        for frame_counter = 1, frame_no - 1 do begin
                              if (pixel_array[cell_counter,frame_counter] gt
(cell_dff_mean + (threshold * cell_dff_stddev))) then begin
                                    yes_no_values[cell_counter, frame_counter] =
1
                              endif
                        endfor
                  endfor
            ;; negative spikes is pos_or_neg = 1
            endif else begin
                  for cell_counter = 0, cell_no - 1 do begin
                        cell_stats = moment(pixel_array[cell_counter,*],
sdev=cell_dff_stddev)
                        cell_dff_mean = cell_stats[0]
                        for frame_counter= 1, frame_no - 1 do begin
                              if (pixel_array[cell_counter,frame_counter] lt
(cell_dff_mean - (threshold * cell_dff_stddev))) then begin
```

```
                                    yes_no_values[cell_counter, frame_counter] =
1
                               endif
                          endfor
                     endfor
               endelse

               spikes_defined = 1

               mess=WIDGET_MESSAGE('Spikes have been found!', /INFORMATION)

               WIDGET_CONTROL, rms_spikes_state.rms_spikes_base, /DESTROY
         endif
end
;-------------------------------------------------------------------------
; NAME:
;       SCORREL_MAP
; PURPOSE:
;       This procedure creates a correlation map between the input cells. A
correlation i,
;     a circular representation of correlation between cells of the slice. Those
cells that
;     are correlated in either direction are joined by lines which are
proportional to their
;     correlation coefficients
; PARAMETERS:
;     symbols:  this is the array of names of companies.
;     output or coef_array:  this is the correlation coeff. array

pro scorrel_map, symbols, output

window, /free, xsize=500, ysize=500, title='Correlation Map Between Stocks'

number=n_elements(symbols)
;_____________________DRAWS CIRCLE OF THE
CELLS_____________________________________________
      radius=5*(number^2)                    ;through trial and error, this seems
the best radius
      offset1=fltarr(number+1)
      offset2=fltarr(number+1)
      offset3=fltarr(number+1)
      offset4=fltarr(number+1)
            plot, fltarr(10), $      ;need to call this blank plot procedure
just to set the ranges for the xyouts
            xrange=[-8*number^2, 8*number^2], $
            yrange=[-8*number^2, 8*number^2], $
            xstyle=4, ystyle=4, xmargin=[0,0], ymargin=[0,0]
            erase                                                ;this is so you
don't see any junk left over..
      FOR I=0, NUMBER-1 DO BEGIN
            ANGLE= ((!pi*2)/number)*i
            offset1(i)=cos(angle)
            offset2(i)=sin(angle)
            offset3(i)=(radius*1.25)*cos(angle)
            offset4(i)=(radius*1.25)*sin(angle)
            xyouts, 1.1*radius*offset1(i), 1.1*radius*offset2(i), symbols(i),
/alignment
```

```
      endfor

;_____________________________PRINT
LINES_____________________________________________________

      ;output_array=print_chi(cut)
      ;this part prints the lines between highly correlated pairs
      ;this is done since the output we have has an empty cell at postion 7.
      ;this finds the cell numbers from thier positions
      for s=0, number-1 do begin
            for t=0, number-1 do begin
                  ;if (output(s, t) gt 0) AND (output(s,t) ne 1) then begin
                  if (output(s,t) gt 0) then begin
               plots, [radius*offset1(s),radius*offset1(t)],
[radius*offset2(s),radius*offset2(t)], thick=3*output(s, t), linestyle=2
 ;;            ;arrow, offset1(s), offset2(s), offset1(t), offset2(t), /data,
thick=3*array(s, t)
                        endif
            endfor
      endfor
;!p.font=0
end
```

```
pro summed_spikes
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size

      base=WIDGET_BASE(/column, title='Overall Stock Behavior')
      sigma=CW_FIELD(base, title='Enter the value for sigma (smoothing
factor):', VALUE=5, $
            UVALUE=1, /RETURN_EVENTS, /FLOATING)
      button1=widget_button(base, value='PLOT', uvalue='plot_pressed')
      WIDGET_CONTROL, base, /realize
      widget_control, base, set_uvalue=sigma

      xmanager, 'summed_spikes', base
end
```

```
pro summed_spikes_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size

      WIDGET_CONTROL, event.top, get_uvalue=sigma
      WIDGET_CONTROL, sigma, GET_VALUE=sigma
      window, 8 ,title='Stock Behavior'

      !except=0
      summed_array=total(yes_no_values,1)
      h=dblarr(frame_no)
      j=dblarr(frame_no)
      z=dblarr(frame_no)
```

```
      for t=0., (frame_no -1), 1 do begin
            h[t]=0
            j[t]=0

            for s=0,(frame_no -1) do begin
                  for i=1,Summed_array[s] do begin                    ;the
two below divisions are unnecessary
                        h[t]=h[t]+(exp(1))^(((-.5 * (s - t)^2) ) / sigma^2)  /
(Sqrt(2 * !pi * sigma^2))
                  endfor
            endfor

            for i=0, (frame_no-1) do begin
                  j[t]=j[t]+(exp(1))^(((-.5 * (i - t)^2) ) / sigma^2)  /
(Sqrt(2 * !pi * sigma^2))
            endfor

            z[t]=h[t]/j[t]
      endfor

      plot,findgen(frame_no)*time_resolution,z, title= 'Plot of overall stock
behavior', $
      xtitle = 'Time (days)', ytitle ='h'
      !except=1
end
```

```
; Name:  two_cells_many_times.pro
; Description: This procedure looks at the cells that fire together more than
once.  It then counts the number of times that pairs
; of cells have fired more than once together.  It also does this for random
cells that have been created through the procedure
; RANDOM_TEST.  The random cells are tested for multiple hits through the
procedure MULTIPLE_TEST_SIGNIFICANCE.
; Explanation of Variables:  no_iterations is the number of iterations the
program will loop to create the random distribution.
;                     least_no_of_matches is the fewest number of times two
cells have to spike together to begin counting the correlation.
;                      window_size is how far, to left and to right, of a spike
we look for other spikes for two cells to be considered coactive

pro two_cells_many_times, least_no_of_matches, no_iterations, window_size

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_create_dist, cells_active                    ;this contains the
number of active cells
forward_function multiple_test_significance
common with_two, filename
common stockdata, symbol_array, date_array

      connections_array = bytarr(cell_no, cell_no)
      yes_no_significance = yes_no_values

      ;initializing the counters to zero
      true_hits = 0
```

```
      ;creating the random distribution
      no_spikes = total(yes_no_values)
      random_distribution = intarr(no_iterations)                    ;this array
holds one random number of matches for every iteration
      for t = 0, no_iterations - 1 do begin                                    ;to
create the random dist. we have to repeat the 2 steps no_iterations times
            make_random_data, seed, random_array, num_active_cells
            random_distribution(t) = count_random_hits_2_manyX(random_array,
least_no_of_matches, window_size, num_active_cells)
      endfor                                                                   ;makes
an array with a number for each iteration
                                                                               ;the
number corresponds to the number of times 2 cells
                                                                               ;fire at
least 'least_no_of_times' in the random movie for the t-th iteration

      ;printing some of the data
      filename=pickfile(/write, file=('Two_Stocks_' +
strcompress(string(least_no_of_matches)) + '_Times'))
      if (filename eq '') then begin
            mess=WIDGET_MESSAGE('This data will not be saved!', /INFORMATION)
            filename='Two_Many.dat'
      endif
      close, 1
      openw, 1, filename
      printf, 1, 'Statistical Data:-'
      printf, 1, 'Number of times two stocks must spike together to count as a
hit:', least_no_of_matches
      printf, 1, 'Number of iterations:          ', no_iterations
      printf, 1, 'Total number of active stocks: ', cells_active

      ;finding the actual number of matches
      distances_array = [0.0]
      for cell_1_counter = 0,cell_no - 2 do begin
            for cell_2_counter = cell_1_counter + 1, cell_no - 1 do begin
                  temp_hits = 0
                  temp_locations_cell_1 = [-1]
                  temp_locations_cell_2 = [-1]
                  for window_counter = -window_size, window_size do begin
                        temp_cell_1 = intarr(frame_no + (2 * window_size))
                        temp_cell_2 = intarr(frame_no + (2 * window_size))
                        temp_cell_1(window_size:frame_no+window_size-1) =
yes_no_values(cell_1_counter, *)
                        temp_cell_2(window_size:frame_no+window_size-1) =
yes_no_values(cell_2_counter, *)
                        temp_cell = (temp_cell_1 * shift(temp_cell_2,
window_counter))
                        ;;;temp_cell = (yes_no_values(cell_1_counter, *) *
shift(yes_no_values(cell_2_counter, *), window_counter))
                        if ((size(where(temp_cell)))[0] eq 1) then begin
                              temp_hits = (temp_hits + total(temp_cell))
                              temp_locations_cell_1 =
[temp_locations_cell_1,where(temp_cell) - window_size]
                              temp_locations_cell_2 =
[temp_locations_cell_2,where(temp_cell) - window_counter - window_size]
                        endif
                  endfor
```

```
                    if (temp_hits ge least_no_of_matches) then begin
                          ;; rearrange location arrays:
                          ;; ...we want to first get rid of all -1 elements
(meaning no paired spikes), then sort the array, then (as sorting returns array
subscripts
                          ;;  and not actual elements) we need to determine what
the original values for the new subscripts are. If you're still confused,
                          ;;  run this line of code in sections on test data, e.g.
'a = [-1,1,2,-1,4,8,-1,5] & a(sort(a(where(a ge 0))))'.
                          real_locations_cell_1 =
(temp_locations_cell_1(where(temp_locations_cell_1 ge
0)))(sort(temp_locations_cell_1(where(temp_locations_cell_1 ge 0))))
                          real_locations_cell_2 =
(temp_locations_cell_2(where(temp_locations_cell_2 ge
0)))(sort(temp_locations_cell_2(where(temp_locations_cell_2 ge 0))))

                          ;; compute connection distance
                          connection_distance =
sqrt(((double(location(cell_1_counter).coord[0]) -
double(location(cell_2_counter).coord[0])) ^ 2) +
((double(location(cell_1_counter).coord[1]) -
double(location(cell_2_counter).coord[1])) ^ 2))
                          distances_array = [distances_array,connection_distance]
                          connections_array(cell_1_counter, cell_2_counter) = 1
                          true_hits = true_hits + 1
                          ;; dump to file
                          printf, 1, 'Stock number one and location:    ',
strcompress(cell_1_counter + 1), location(cell_1_counter).coord
                          printf, 1, 'Frames cell one spikes in:'
                          printf, 1, real_locations_cell_1
                          printf, 1, 'Stock number two and location:    ',
strcompress(cell_2_counter + 1), location(cell_2_counter).coord
                          printf, 1, 'Frames cell two spikes in:'
                          printf, 1, real_locations_cell_2
                          printf, 1, ' '
                          ;; update 'significance array' which is used to draw the
raster plot
                          for frame_counter = 0, temp_hits - 1 do begin
                               yes_no_significance(cell_1_counter,
real_locations_cell_1[frame_counter])=TEMPORARY(yes_no_significance(cell_1_count
er, real_locations_cell_1[frame_counter])) + 1
                               yes_no_significance(cell_2_counter,
real_locations_cell_2[frame_counter])=TEMPORARY(yes_no_significance(cell_2_count
er, real_locations_cell_2[frame_counter])) + 1
                          endfor
                    endif
               endfor
          endfor

          ;draw raster plot before it checks to see if the random distribution has a
variance as the raster is independant of statistics
          draw_significance_raster, yes_no_significance

          ;check to see if event occurs randomly and thus if a p value can be
calculated.
          xmax=max(random_distribution)
```

```
      xmin=min(random_distribution)

      if (xmax eq xmin) then begin
            mess=WIDGET_MESSAGE('Random distribution has variance of zero.  Try
 again with greater number of iterations', /error)
            printf, 1, 'ERROR!'
            printf, 1, 'Moment undefined for random distribution with variance
 zero'
            free_lun, 1
            close, 1
            p_value=-1
      endif else begin
            ;making the histogram for the random distribution
            !p.multi=0
            bin_size=1
            hist=histogram(random_distribution, binsize=1, min=0,
 max=(2*xmax)+1)         ;plot a histogram of the distribution
            window, /free, title='Two Many Distribution'
            plot, hist, xtitle='number of hits', ytitle='frequency'
            y2=total(random_distribution eq true_hits)      ;for drawing the
 blue line for the real data, we have to find the height of that line in the
histogram
            if y2 le 0 then begin                        ;if there are no random
values equal to the true data, draw a line of height one
                  y2=1
            endif
            plots, [true_hits, true_hits], [0, y2], color=12      ;this draws a
line where the actual number of matches lies

            ;calculating the number of spikes per cell per second

      spikes_per_cell_per_second=(no_spikes/cells_active)/(frame_no*time_resolut
ion)

            ;draw the correlation map
            connections_array=(connections_array)+TRANSPOSE(connections_array)
      ;this is to make the connections_array symmetric. Because if cell 1 is
 connected to 2, 2 is also connected to 1. correl_map_plane needs the symmetric
 array
            scorrel_map, symbol_array, connections_array
            ;;window, /free, xsize=x_size, ysize=y_size, title='Correl Map for
 Two Many'
            ;;correl_map_image_plane, connections_array

            ;find the p value, standard deviation etc.
            stats=moment(random_distribution, sdev=sdev)
            no_points_right=total(random_distribution ge true_hits)
            p_value=no_points_right/no_iterations

            ;print the data to a file
            printf, 1, 'Total number of frames:       ', strcompress(frame_no)
            printf, 1, 'Total number of spikes:       ', strcompress(no_spikes)
            printf, 1, 'Mean expected matches:        ', strcompress(stats(0))
            printf, 1, 'Variance:                           ',
 strcompress(stats(1))
            printf, 1, 'Standard deviation:                 ', strcompress(sdev)
```

```
            printf, 1, 'Actual matches:                ', strcompress(true_hits)
            printf, 1, 'Actual/expected:               ',
strcompress(true_hits/stats(0))
            printf, 1, 'Standard error for ratio:          ',
strcompress(sdev/stats(0))
            printf, 1, 'Spike firing rate:                 ',
strcompress(spikes_per_cell_per_second)
            if (n_elements(distances_array) gt 2) then begin
                  distance_stats =
moment(distances_array[1:n_elements(distances_array)-1],
sdev=sdev_distances_array)
                  printf, 1, 'Mean connection distance:          ',
strcompress(distance_stats[0])
                  printf, 1, 'Standard deviation:                ',
strcompress(sdev_distances_array)
            endif else begin
                  if (n_elements(distances_array) gt 1) then begin
                        printf, 1, 'Connection distance:          ',
strcompress(distances_array[1])
                  endif
            endelse
            printf, 1, 'Significance p-value:         ', strcompress(p_value)
            close, 1
      endelse
end
```

```
;     Name: two_many_widget
;     Description: Using this widget the user can test the significance of two
cells firing many times in
;           the movie to analyze.  The user will have to input the
specifications such as the number of times
;           that two cells fire together to be taken as the min. criterion for a
hit.

pro two_many_widget

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size

      two_many_base=WIDGET_BASE(/COLUMN, title='Significance of two stocks
spiking together many times')
      field1=CW_FIELD(two_many_base, /RETURN_EVENTS, /INTEGER, TITLE='Two stocks
should spike together at least this many times:', VALUE=2, UVALUE=0)
      field2=CW_FIELD(two_many_base, /RETURN_EVENTS, /INTEGER, TITLE='Number of
iterations:', VALUE=1000, UVALUE=0)
      field3=CW_FIELD(two_many_base, /RETURN_EVENTS, /INTEGER, TITLE='Window
size for hits:', VALUE=0, UVALUE=0)
      button1=WIDGET_BUTTON(two_many_base, VALUE='Find significance', UVALUE=2)
      WIDGET_CONTROL, /realize, two_many_base
      two_many_state={field1:field1, field2:field2, field3:field3,
button1:button1}
      WIDGET_CONTROL, WIDGET_INFO(two_many_base, /CHILD),
SET_UVALUE=two_many_state
      xmanager, 'two_many_widget', two_many_base
end
```

```
;     NAME: two_many_widget_event

pro two_many_widget_event, event
common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
common with_two, filename

      two_many_state_info=WIDGET_INFO(event.handler, /CHILD)
      WIDGET_CONTROL, two_many_state_info, GET_UVALUE=two_many_state
      WIDGET_CONTROL, two_many_state.field1, GET_VALUE=least_no_of_matches
      WIDGET_CONTROL, two_many_state.field2, GET_VALUE=no_of_iterations
      WIDGET_CONTROL, two_many_state.field3, GET_VALUE=window_size

      if (event.id eq two_many_state.button1) then begin
            if ((least_no_of_matches le 0) or (no_of_iterations le 1) or
(window_size lt 0) or (window_size gt frame_no)) then begin
                  mess=WIDGET_MESSAGE('Invalid fields specified!', /ERROR)
            endif else begin
                  two_cells_many_times, least_no_of_matches, no_of_iterations,
window_size
                  WIDGET_CONTROL, event.top, /hourglass
                  close, 1
                  openr, 1, filename
                  stat_stuff=fstat(1)
                  file_size=stat_stuff.size
                  close, 1
                  if (file_size gt 8112) then begin
                        mess=WIDGET_MESSAGE('File is too large to display
through a widget. Open it manually. If you did not save it, it is named
Two_Many.dat', /INFORMATION)
                  endif else begin
                        xdisplayfile, filename, title = "Statistical Data for
Two Many", group = event.top, width = 75, height = 50
                  endelse
            endelse
      endif
end
```

```
; Widget for creating correlation coefficients matrix and drawing general
correlation maps.

pro widget_analyze

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
      total_frame_no, time_resolution, x_size, y_size, box_size
      base1=WIDGET_BASE(/COLUMN, title='General Slice Correlation')
      type={cw_pdmenu_s, flags:0, name:''}
      nitty_gritty =[   { cw_pdmenu_s, 0, 'Calculate Correlation Coeff Matrix'},
$
            { cw_pdmenu_s, 1, 'Build Correlation Map'}, $
            { cw_pdmenu_s, 0, 'On Image'}, $
            { cw_pdmenu_s, 2, 'Spatial'}]
      pull_down2=CW_PDMENU(base1, nitty_gritty, /RETURN_FULL_NAME)

      WIDGET_CONTROL, base1, /realize
      xmanager, 'widget_analyze', base1
```

```
end

; Name: widget_analyze_event
; Description: This procedure is called when you press the 'Analyze' button on
the main menu.
; It allows you to calculate the correlation coefficient (WHICH MUST BE THE
FIRST STEP) and then
; draw a correlation map on an image or spatial.

pro widget_analyze_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
total_frame_no, time_resolution, x_size, y_size, box_size
common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined

   CASE (event.value) of
            'Calculate Correlation Coeff Matrix': BEGIN
                  choose_correl
              END

            'Build Correlation Map.On Image': BEGIN
            if (correl_coef_defined ne 1) then begin
                  mess=WIDGET_MESSAGE('You have to first calculate the
correlation coefficients!', /Error)
            endif else begin
                     scorrel_map
            endelse
            END

            'Build Correlation Map.Spatial': BEGIN
              if (correl_coef_defined ne 1) then begin
                  mess=WIDGET_MESSAGE('You have to first calculate the
correlation coefficients!', /Error)
            endif else begin
                  base5=WIDGET_BASE(/COLUMN, title='Spatial Correlation Map')
                  draw7=WIDGET_DRAW(base5, xsize=x_size, ysize=y_size)
                  WIDGET_CONTROL, /realize, base5
                  correl_map_image_plane, coef
            endelse
             END
  ENDCASE
end
```

APPENDIX B

```
; to find argument of the maximum value of the array
; used by hidden_markov_event.pro

function arg_max, x,y,j,t,D,A

Z=dblarr(y+1)

for i=x,y do begin
if( A[i,j] eq 0) then begin
z[i]=-10D307
endif else begin
z[i] = D[t-1,i]+alog(A[i,j])
endelse
endfor

m=max(Z,k)
return, k
end
```

```
; function used by HMM to find the probability of observation vector of
; length cell_no, at time t, given the array of Poisson lambdas B, and state_no

function B_prob, state_no, observation, t, B, cell_no

;!except=2
s=double(1)

for i=0, cell_no -1 do begin
observation(i,t)=double(observation(i,t))
 if (B[state_no,i] eq 0) then begin
if (observation(i,t) eq 0) then begin
s=s * 1.
endif else begin
s=s * 0
endelse
endif else begin
s=s*(  (exp(-B[state_no,i]))  *  ((B[state_no,i])^(observation(i,t)))  /
factorial(observation(i,t))   )
;s=s - B[state_no,i] + (observation(i,t))*alog(B[state_no,i]) -
alog(factorial(observation(i,t)))
endelse
endfor
;s=exp(s)
return, s
end
```

```
pro choose_init_par


common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov,
n,m,A,B,P,observation,f,g,d,Fi,Q,back,xsi,Gamma,n_i,a_i,b_i,p_i,c,w_s,t_max
common indexes, i_a,i_b,i_p
```

```
i_a=0
i_b=0
i_p=0

Values=[ 'random','uniform']
Values2=['random', 'uniform','averaged over intervals']
base=widget_base(/column, title='Initial parameters generation')

Ainit=widget_droplist(base, value=values, uvalue=0, title='Choose initial state
transition prob. A:  ')
Binit=widget_droplist(base, value=values2, uvalue=1, title='Choose initial state
characteristcs B:    ')
Pinit=widget_droplist(base, value=values, uvalue=2, title='Choose initial state
prob. P:             ')
Apply=widget_button(base, value='Apply', uvalue=3)

widget_control, base, /realize
xmanager,'choose_init_par', base
end
```

```
pro choose_init_par_event, event


common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov,
n,m,A,B,P,observation,f,g,d,Fi,Q,back,xsi,Gamma,n_i,a_i,b_i,p_i,c,w_s,t_max
common indexes, i_a,i_b,i_p

WIDGET_CONTROL, event.id, GET_UVALUE=uval


IF (TAG_NAMES(event, /STRUCTURE_NAME) EQ 'WIDGET_DROPLIST') $
  THEN BEGIN
    Case uval of
    0: begin
            CASE event.index OF

            0: begin
            ;print, 'A random'
            i_a=i_a+1
            ;Random distribution
            for i=0, (n-1) do begin
            for j=0, (n-1) do begin
            A[i,j]=randomu(seed)
            endfor
            endfor
            z1=dblarr(n)
            z1=total(A,2)
            for i=0, (n-1) do begin
            for j=0, (n-1) do begin
            A[i,j]=A[i,j]/z1[i]    ;?????
            endfor
            endfor
            end
```

```
            1: begin
            ;print, 'A uniform'
            i_a=i_a+1
            ;Uniform distribution
            for i=0, (n-1) do begin
            for j=0, (n-1) do begin
            A[i,j]=1/n
            endfor
            endfor
            end

            ENDCASE
        end
    1: begin
        CASE event.index OF

            0: begin
            ;print, 'B random'
            i_b=i_b+1
            for i=0, (n-1) do begin
            for j=0, (m-1) do begin
            B[i,j]=Randomu(seed)   ;Random
            endfor
            endfor
            z1=total(b,2)
            for i=0, (n-1) do begin
            for j=0, (m-1) do begin
            B[i,j]=B[i,j]/z1[i]      ;think
            endfor
            endfor
            end

            1: begin
            ;print, 'B uniform'
            i_b=i_b+1
            ; uniform
            for i=0, (n-1) do begin
            for j=0, (m-1) do begin
            ;mm=m.
            B[i,j]=1/float(m)
            endfor
            endfor
            end

            2: begin ; average over uniform segments
            i_b=i_b+1
            for j=0, m-1 do begin
            for i=0, n-1 do begin
            s=0
            s_i=0
            for k=fix(i*((frame_no - w_s + 1)/n)) , fix((i+1)*((frame_no -w_s +
1)/n)-1) do begin  ;change frame_no
            s=s+observation[j,k]
            s_i=s_i+1
            ;print, k
            endfor
            ;print, 'dupa'
```

```
            s=double(s)
            s_i=double(s_i)
            B[i,j]=double( s / s_i)
            endfor
            endfor
            end

            ENDCASE
       end

   2: begin
       CASE event.index OF

            0: begin
            ;print, 'P random'
            i_p=i_p+1
            for i=0, (n-1) do begin
            P(i)=randomu(seed) ;random
            endfor
            z2=total(p)
            for i=0, (n-1) do begin
            P(i)=P(i)/z2
            endfor
            end

            1: begin
            ;print, 'P uniform'
            i_p=i_p+1
            ; uniform
            P(0)=1
            for i=1, (n-1) do begin
            P(i)=0
            endfor
            end
            ENDCASE
       end
    endcase
ENDIF


if (uval eq 3) then begin
print, 'Apply pressed'
;print, 'i_a=',i_a,', i_b=',i_b,', i_p=',i_p

if(i_a eq 0) then begin ;make random
            ;print, 'A random'
            for i=0, (n-1) do begin
            for j=0, (n-1) do begin
            A[i,j]=randomu(seed)
            endfor
            endfor
            z1=dblarr(n)
            z1=total(A,2)
            for i=0, (n-1) do begin
            for j=0, (n-1) do begin
            A[i,j]=A[i,j]/z1[i]    ;?????
            endfor
```

```
            endfor
endif
if(i_b eq 0) then begin ; make random
            ;print, 'B random'
            for i=0, (n-1) do begin
            for j=0, (m-1) do begin
            B[i,j]=Randomu(seed)
            endfor
            endfor
            z1=total(b,2)
            for i=0, (n-1) do begin
            for j=0, (m-1) do begin
            B[i,j]=B[i,j]/z1[i]      ;think
            endfor
            endfor
endif
if(i_p eq 0) then begin ; make random
            ;print, 'P random'
            for i=0, (n-1) do begin
            P(i)=randomu(seed)
            endfor
            z2=total(p)
            for i=0, (n-1) do begin
            P(i)=P(i)/z2
            endfor
endif

print, 'These are initial state transition probabilities, A'
print, A
print, 'space'
print, 'These are initial probabilities of observing symbol m at state n, B'
print, b
print, 'space'
print, 'This is the initial state distribution, P'
print, p
print, 'space'

n_i=n ;for save function
A_i=A
B_i=B
P_i=P
widget_control, event.top, /destroy
endif
end
```

```
pro hidden_markov


common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov,
n,m,A,B,P,observation,f,g,d,Fi,Q,back,xsi,Gamma,n_i,a_i,b_i,p_i,c,w_s,t_max
common markov2, yes_no_values3,yes_no_temp,cell_no_temp,frame_no_temp



; this is the definition of a test input with 400 frames and 4 states
```

```
;frame_no=400
;observation=intarr(frame_no) ;observation sequence
;for i=0,99 do begin
;x=randomn(seed,binomial=[1,.3])
;if (x eq 1) then begin
;observation[i]=1
;endif else begin
;observation[i]=2
;endelse
;endfor
;for i=100,199 do begin
;x=randomn(seed,binomial=[1,.6])
;if (x eq 1) then begin
;observation[i]=1
;endif else begin
;observation[i]=3
;endelse
;endfor
;for i=200,299 do begin
;x=randomn(seed,binomial=[1,.1])
;if (x eq 1) then begin
;observation[i]=2
;endif else begin
;observation[i]=0
;endelse
;endfor
;for i=300,399 do begin
;x=randomn(seed,binomial=[1,.5])
;if (x eq 1) then begin
;y=randomn(seed,binomial=[1,.4])
;if(y eq 1) then begin
;observation[i]=1
;endif else begin
;observation[i]=2
;endelse
;endif else begin
;y=randomn(seed,binomial=[1,.4])
;if(y eq 1) then begin
;observation[i]=0
;endif else begin
;observation[i]=3
;endelse
;endelse
;endfor
;m=double(4)



;this transforms yes_no_values to observation seq., information is lost
;because only one cell per time unit is allowed to spike
;observation=intarr(frame_no) ;observation sequence
;for i=0,(frame_no -1) do begin
;observation[i]=0
;for j=0, (cell_no -1) do begin
;if (yes_no_values[j,i] eq 1) then begin
;observation[i]=j+1      ;needs change
;endif
```

```
;endfor
;endfor
;m=double(cell_no + 1) ; number of observation symbols (+1 if neither cell fires
- 0)



;this transforms yes_no_values to observation by adding everything - creates
artificially many zeros


;m=double(cell_no + 1) ; number of observation symbols (+1 if neither cell fires
- 0)
;observation1=intarr(frame_no * cell_no) ;observation sequence
;z=0
;for j=0, (frame_no -1) do begin
;for i=0, (cell_no -1) do begin
;if(yes_no_values[i,j] eq 1) then begin
;observation1[z]=i+1
;endif else begin
;observation1[z]=0
;endelse
;z=z+1
;endfor
;endfor
;frame_no=frame_no*cell_no


;this creates observation from yes-no_values by adding everything but only if
cells fire simulatneously
;m=double(cell_no + 1) ; number of observation symbols (+1 if neither cell fires
- 0)
;observation_2=intarr(frame_no * cell_no) ;observation sequence
;z=0
;for j=0, (frame_no -1) do begin
;z2=0
;for i=0, (cell_no -1) do begin
;if(yes_no_values[i,j] eq 1) then begin
;observation_2[z]=i+1
;z=z+1
;z2=1
;endif else begin
;observation_2[z]=0
;endelse
;endfor
;if(z2 eq 0) then begin
;z=z+1
;endif
;endfor

;observation2=intarr(z)
;for i=0,(z-1) do begin
;observation2[i]=observation_2[i]
;endfor
;frame_no=z
```

```
;observation=observation2



;test=fltarr(2,3)
;test[0,0]=1.1
;test[0,1]=1.2
;test[0,2]=1.3
;test[1,0]=2.1
;test[1,1]=2.2
;test[1,2]=2.3
;sum1=fltarr(5)
;sum1=total(test,1)
;sum2=fltarr(5)
;sum2=total(test,2)
;print, test
;print, 'this is sum 1'
;print, sum1
;print, 'this is sum 2'
;print, sum2


base=widget_base(/column, title='Hidden Markov Modelling')
stuff= {cw_pdmenu_s, flags:0, name:''}
details=[{ cw_pdmenu_s, 1, 'Create input data' }, $
        { cw_pdmenu_s, 0, 'from observation sequence'},$
        { cw_pdmenu_s, 0, 'from shuffled in time observation sequence'},$
        { cw_pdmenu_s, 0, 'from Monte Carlo simulated observation seq (space)'},$
        { cw_pdmenu_s, 2, 'test'},$
        { cw_pdmenu_s, 0, 'Record y_n_values'},$
        { cw_pdmenu_s, 0, 'Retrieve y_n_values'}]
pull_down=cw_pdmenu(base, details, /return_name, UVALUE=4)
button1=cw_bgroup(base, /row, ['Choose initial parameters.', 'Save init. pars.',
'Load init. pars.', 'Save final pars.', 'Load final pars.'], /return_name,
UVALUE=0)
state_no=CW_FIELD(base, title='Enter the number of states you would like to use
in the model:', VALUE=4, $
            UVALUE=5, /RETURN_EVENTS,/Floating)
button2=widget_button(base, value='Estimate model parameters.', uvalue=3)
iteration_no=CW_FIELD(base, title='Enter the number of iterations you would like
to perform:', VALUE=20, $
            UVALUE=4, /RETURN_EVENTS, /FLOATING)
button3=widget_button(base, value='Find the most probable hidden state
sequence.', uvalue=2)
button4=widget_button(base, value='Compute P(observation | model parameters)',
uvalue=1)
button5=widget_button(base, value='Find cross correlations within states',
uvalue=6)
correlation_state=CW_FIELD(base, title='In which state do you want to find
cross-correlations ?:', VALUE=0, $
            UVALUE=7, /RETURN_EVENTS, /FLOATING)
state={state_no:state_no,iteration_no:iteration_no,
correlation_state:correlation_state}
widget_control, base, set_uvalue=state
```

```
WIDGET_CONTROL, state_no, GET_VALUE=n
widget_control, base, /realize

xmanager,'hidden_markov', base
end

---------------------------------------------------------------------
pro hidden_markov_event, event

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov,
n,m,A,B,P,observation,f,g,d,Fi,Q,back,xsi,Gamma,n_i,a_i,b_i,p_i,c,w_s,t_max
common markov2, yes_no_values3, yes_no_temp,cell_no_temp,frame_no_temp

;!except=2
n_old=n
WIDGET_CONTROL, event.id, GET_UVALUE=uval
Widget_control, event.top, get_uvalue=state
WIDGET_CONTROL, state.iteration_no, GET_VALUE=iteration_no
WIDGET_CONTROL, state.state_no, GET_VALUE=n
WIDGET_CONTROL, state.correlation_state, GET_VALUE=correlation_state


CASE uval OF

4: Begin

Case event.value OF

'from observation sequence': BEGIN

;print, 'these are y_n_values'
;print, yes_no_values
;print,''

m=cell_no
w_s= 30. ; size of sliding  window, step is equal to 1
observation=intarr(cell_no, FRAME_NO -  w_s +1)
for i=0, (frame_no - w_s ) do begin
for k=0, (cell_no -1) do begin
s=0
if(i lt (frame_no - w_s +1)) then begin
for j=i, (i+w_s -1) do begin
s=s+yes_no_values[k,j]
endfor
observation[k,i]=s
endif
endfor
endfor
print, 'Input from observation sequence created'

end


'from shuffled in time observation sequence': BEGIN
```

```
yes_no_values2=intarr(cell_no, frame_no)
shuffle=intarr(cell_no)

for i=0, cell_no -1 do begin
shuffle[i]=(frame_no / 2) * randomn(seed, uniform=1)
endfor

for i=0, cell_no -1 do begin
for j=0, frame_no -1 - shuffle[i] do begin
yes_no_values2[i,j]=yes_no_values[i,j+shuffle[i]]
endfor
for j=(frame_no - shuffle[i]), (frame_no -1) do begin
yes_no_values2[i,j]=yes_no_values[i,shuffle[i] - frame_no + j]
endfor
endfor

;print,'These are yes and no values:'
;print, yes_no_values2
;print,''
;yes_no_values=yes_no_values2

m=cell_no
w_s= 30. ; size of sliding  window, step is equal to 1
observation=intarr(cell_no, FRAME_NO -  w_s +1)
for i=0, (frame_no - w_s ) do begin
for k=0, (cell_no -1) do begin
s=0
if(i lt (frame_no - w_s +1)) then begin
for j=i, (i+w_s -1) do begin
s=s+yes_no_values[k,j]
endfor
observation[k,i]=s
endif
endfor
endfor
print, 'Input from shuffled in time observation sequence created'

end


'from Monte Carlo simulated observation seq (space)': BEGIN  ;this is copied
from vikram's make_random_data.pro


      ;find the number of spikes each cell of the true data has
      no_spikes=intarr(cell_no)       ;this is the array with the number of
spikes for each cell
      for cell=0, cell_no-1 do begin
            no_spikes(cell)=total(yes_no_values(cell, *))
      endfor
      cells_active=total(no_spikes gt 0)
      ;create a random binary spike train.  First, numbers for locations of
spikes are generated randomnly making
      ;sure that no number is repeated more than once.  Next, a binary spike
train is created using these locations
```

```
      ;for the position of spikes. Each cell has the same number of spikes as
the true data had.

      random_array=intarr(cell_no, frame_no)
      for cell=0, cell_no-1 do begin
            index=no_spikes(cell)-1

            ;using a random number generator for a uniform distribution
            ;here we see if there are any spikes in the original cell of the
data
            if index ge 0 then begin
                  repeat begin
                        temp=randomu(my_seed, no_spikes(cell))          ;random
numbers
                        temp=fix(frame_no*temp)
                        ;this has the location of the unique elements in the
random row of data
                        ;this is the test to make sure no numbers are repeated
in the data
                        unique=(uniq(temp(sort(temp))))                 ;# of
unique elements
                        number1=n_elements(unique)                ;total number
of elements
                        number2=n_elements(temp)
                  endrep until number1 eq number2

                  random_array(cell, temp-1)=1                    ;put spikes at
the random locations
            endif
      endfor
yes_no_values=random_array
;print,'this are y_n values'
;print, yes_no_values

m=cell_no
w_s= 30. ; size of sliding  window, step is equal to 1
observation=intarr(cell_no, FRAME_NO -  w_s +1)
for i=0, (frame_no - w_s ) do begin
for k=0, (cell_no -1) do begin
s=0
if(i lt (frame_no - w_s +1)) then begin
for j=i, (i+w_s -1) do begin
s=s+yes_no_values[k,j]
endfor
observation[k,i]=s
endif
endfor
endfor

print, 'Input from monte carlo simulated data created'

end

'test': BEGIN
; this is going to be a test

print, 'Frames 1-100    should be in state 0 with paramters: [ 3,2,8,1]'
```

```
print, 'Frames 101-200 should be in state 1 with parameters: [ 3,4,8,4]'
print, 'Frames 201-300 should be in state 2 with parameters: [ 3,3,6,0]'
print, 'Frames 301-400 should be in state 3 with parameters: [11,3,6,4]'



m=4.
cell_no=4.
w_s=11
frame_no=410
observation=intarr(cell_no, FRAME_NO -  w_s +1)
for i=0, (99 ) do begin
observation[0,i]=randomn(seed, poisson=3)
observation[1,i]=randomn(seed, poisson=2)
observation[2,i]=randomn(seed, poisson=8)
observation[3,i]=randomn(seed, poisson=1)
endfor
for i=100, (199 ) do begin
observation[0,i]=randomn(seed, poisson=3)
observation[1,i]=randomn(seed, poisson=4)
observation[2,i]=randomn(seed, poisson=8)
observation[3,i]=randomn(seed, poisson=4)
endfor
for i=200, (299 ) do begin
observation[0,i]=randomn(seed, poisson=3)
observation[1,i]=randomn(seed, poisson=3)
observation[2,i]=randomn(seed, poisson=6)
observation[3,i]=0
endfor
for i=300, (399 ) do begin
observation[0,i]=randomn(seed, poisson=11)
observation[1,i]=randomn(seed, poisson=3)
observation[2,i]=randomn(seed, poisson=6)
observation[3,i]=randomn(seed, poisson=4)
endfor
print, 'test input created'

end

'Record y_n_values': begin
yes_no_temp=yes_no_values
cell_no_temp=cell_no
frame_no_temp=frame_no
print, 'Yes and no values recorded'
end



'Retrieve y_n_values': begin
cell_no=cell_no_temp
frame_no=frame_no_temp
yes_no_values=intarr(cell_no,frame_no)
yes_no_values=yes_no_temp
print, 'Yes and no values retrieved'
end

'test2':begin
```

```
for i=0,4 do begin
print, yes_no_values[i,7]
endfor
end

else:
endcase
;print, 'This is the input:'
;print, observation
;print,''

t_max=frame_no - w_s +1
n=double(n)
A=dblarr(n,n)
B=dblarr(n,m)
P=dblarr(n)

end

0: Begin

Case event.value OF

'Choose initial parameters.': BEGIN



if(n_old ne n) then begin
n=double(n)
A=dblarr(n,n)
B=dblarr(n,m)
P=dblarr(n)
endif

choose_init_par
end

'Save init. pars.': BEGIN

            data_file=pickfile(/write, file='HM_IVars_exp', title='Create the
saved variables file')
            if (data_file eq '') then begin
                        mess=WIDGET_MESSAGE('No saved variables file specified',
/INFORMATION)
                    endif else begin
                        Save, FileName=data_file, A_i, B_i, P_i, n_i
                  mess=WIDGET_MESSAGE('Initial data from this experiment saved',
/INFORMATION)
            endelse

end

'Load init. pars.': BEGIN

                        data_file=pickfile(/read, title='Select the saved
variables file', GET_PATH=filepath1)
                        if (data_file eq '') then begin
```

```
                        mess=WIDGET_MESSAGE('No saved variables file specified',
/INFORMATION)
                        endif else begin
                        restore, data_file
                        mess=WIDGET_MESSAGE('Previous initial A, B, P, and n
variables loaded', /INFORMATION)
                    endelse

              data_file=0
              a=a_i
              b=b_i
              p=p_i
              n=n_i
              print, 'This is initial A'
              print, A
              print, ''
              print, 'This is initial B'
              print, B
              print, ''
              print, 'this is initial p'
              print, p
              print, ''


              WIDGET_CONTROL, state.state_no, SET_VALUE=n
end

'Save final pars.': BEGIN

              data_file=pickfile(/write, file='HM_FVars_exp', title='Create the
saved variables file')
              if (data_file eq '') then begin
                        mess=WIDGET_MESSAGE('No saved variables file specified',
/INFORMATION)
                    endif else begin
                        Save, FileName=data_file, A, B, P, n
                   mess=WIDGET_MESSAGE('Data from this experiment saved',
/INFORMATION)
              endelse
end

'Load final pars.': BEGIN

                         data_file=pickfile(/read, title='Select the saved
variables file', GET_PATH=filepath1)
                         if (data_file eq '') then begin
                        mess=WIDGET_MESSAGE('No saved variables file specified',
/INFORMATION)
                        endif else begin
                        restore, data_file
                        mess=WIDGET_MESSAGE('A, B, P, and n variables loaded',
/INFORMATION)
                    endelse

              data_file=0

                print, 'This is  A'
```

```
            print, A
            print, ''
            print, 'This is  B'
            print, B
            print, ''
            print, 'this is  p'
            print, p
            print, ''


            WIDGET_CONTROL, state.state_no, SET_VALUE=n
end


else:
endcase
end

1: Begin


;The Log probability of the observation given the model
g=double(0)
;for i=0, n-1 do begin
;g=g+f(t_max-1,i)
;endfor
for t=0,t_max -1 do begin
g=g - alog(c(t))
endfor
print, 'This is the Log probability of the observation sequence given the model'
print, g
print, ''
end


;2 :begin  ; Viterbi with no scaling
;print, 'button 2 pressed'

; This is an implementation of the viterbi algorithm. the pupose of
; viterbi algorithm is to find the single most probable state sequence given
; the observation sequence and model parameters.

;!except=2
;D=dblarr(t_max,n) ;highest probability along a single state path of the
observation sequence
;Fi=dblarr(t_max,n) ; the backtracking step to retrieve the hidden states

;initiation
;for i=0, n-1 do begin
;D[0,i]=p[i]*B_prob(i,observation,0)
;Fi[0,i]=0
;endfor

;recursion
;Z=dblarr(n)
;for t=1, t_max-1 do begin
;for j=0, n-1 do begin
```

```
;for i=0, n-1 do begin
;Z[i]=D[t-1,i]*A[i,j]
;endfor
;D[t,j]=(max(Z))*B[j,(observation(t))]
;Fi[t,j]=arg_max(0,n-1,j,t,D,A) ; i changes from  0 to n-1
;endfor
;endfor

;termination
;for i=0, n-1 do begin
;Z[i]=D[t_max-1,i]
;endfor
;Prob_path=max(Z,k) ; probability of the given path
;Q=intarr(t_max)    ; Q is the hidden state sequence
;Q[t_max -1]=k
;for t=t_max-2,0,-1 do begin
;Q[t]=Fi[t+1,(Q[t+1])]
;endfor
;print,'This is D'
;print, D
;print, 'this is Fi'
;print, Fi
;print, 'This is Q'
;print, Q
;print, ''
;print, 'This is the corresponding probability of the hidden state sequence
given the observation data:'
;print, Prob_path
;print, ''
;end

2 :begin
;print, 'button 2 pressed'

; This is an implementation of the viterbi algorithm. the pupose of
; viterbi algorithm is to find the single most probable state sequence given
; the observation sequence and model parameters. here i use logarithmic scaling


D=dblarr(t_max,n) ;highest probability along a single state path of the
observation sequence
Fi=dblarr(t_max,n) ; the backtracking step to retrieve the hidden states

;initiation
for i=0, n-1 do begin
if( (P(i) eq 0) or (B_prob(i,observation,0,B,cell_no) eq 0) )then begin
D[0,i]=-10D307
Fi[0,i]=0
endif else begin
D[0,i]=alog(P(i))+alog(B_prob(i,observation,0,B,cell_no))
Fi[0,i]=0
endelse
endfor

;recursion
Z=dblarr(n)
for t=1, t_max-1 do begin
```

```
for j=0, n-1 do begin
for i=0, n-1 do begin
if( A[i,j] eq 0) then begin
Z[i]=-10D307
endif else begin
Z[i]=D[t-1,i]+alog(A[i,j])
endelse
endfor
if ( B_prob(j,observation,t,B,cell_no) eq 0) then begin
D[t,j]=-10D307
Fi[t,j]=arg_max(0,n-1,j,t,D,A) ; i changes from  0 to n-1
endif else begin
D[t,j]=(max(Z))+alog(B_prob(j,observation,t,B,cell_no))
Fi[t,j]=arg_max(0,n-1,j,t,D,A) ; i changes from  0 to n-1
endelse
endfor
endfor

;termination
for i=0, n-1 do begin
Z[i]=D[t_max-1,i]
endfor
ln_of_Prob_path=max(Z,k) ; probability of the given path
Q=intarr(t_max)    ; Q is the hidden state sequence
Q[t_max -1]=k
for t=t_max-2,0,-1 do begin
Q[t]=Fi[t+1,(Q[t+1])]
endfor
print, 'This is Q'
print, Q
print, ''
print, 'This is  LogP of the hidden state sequence given the observation data:'
print, ln_of_Prob_path
print, ''
end

3 :begin
print, 'button 3 pressed'

for i=0, iteration_no do begin
par_reestimation
endfor

print, 'this is new A'
print, A
print, ''
print, 'This is new B'
print, B
print, ''
print, 'this is new P'
print, p
print, ''

end

6: begin
```

```
;cell_no=2
;w_s=3
;frame_no=15
;t_max=frame_no - w_s + 1
;yes_no_values=intarr(cell_no,frame_no)
;yes_no_values[0,*]=findgen(frame_no)
;yes_no_values[1,*]=findgen(frame_no)
;q=intarr(t_max)
;q=[1,0,0,1,0,0,1,0,0,1,0,0,1]
;correlation_state=1

yes_no_values2=intarr(cell_no,frame_no)
for i=0, frame_no -1 do begin
for j=0, cell_no -1 do begin
yes_no_values2[j,i]=-1
endfor
endfor

for i=0, frame_no -1 do begin
OK=0
for j=0, w_s -1 do begin    ;possibly change the j range to make the window
smaller
if(((i-j) ge 0) and ((i-j) lt t_max)) then begin
if( q(i-j) eq correlation_state ) then OK=1
endif
endfor
for k=0, cell_no -1 do begin
if (OK eq 1) then Yes_no_values2[k,i]=yes_no_values[k,i]
endfor
endfor

;print,'frmae number is', frame_no
;print, 't max is', t_max
;print, 'w_s is', w_s


xx=where(yes_no_values2[0,*] eq -1,count)
yes_no_values3=intarr(cell_no,frame_no - count)
for j=0, cell_no -1 do begin
k=0
for i=0, frame_no -1 do begin
if(yes_no_values2[j,i] ne -1) then begin
yes_no_values3[j,k]= yes_no_values2[j,i]
k=k+1
endif
endfor
endfor

;print, 'these are yes no values'
;print, yes_no_values
;print,q
;print, 'these are yes no values 2'
;print, yes_no_values2
;print, 'these are yes no values 3'
;print, yes_no_values3
print, 'State constrained yes and no values created'
```

```
draw_cross2

end
endcase
end
```

```
pro par_reestimation

common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common markov,
n,m,A,B,P,observation,f,g,d,Fi,Q,back,xsi,Gamma,n_i,a_i,b_i,p_i,c,w_s,t_max

t_max=frame_no - w_s +1

;!except=2
;Let's recalculate forward variables: F
;f=dblarr(t_max,n)
;for i=0, n-1 do begin
;f(0,i)=P(i)*B(i,observation(0))
;endfor
;for t=0, (t_max-2) do begin
;for j=0, (n-1) do begin      ; f(t,i),A(i,j)
;f(t+1,j)=sum(0,n-1,t,j,f,A)*B(j,observation(t+1)) ;i goes from 0 to n-1
;endfor
;endfor

;I will apply scaling procedure to prevent underflows and increase accuracy
; c's will be the scaling coefficients
c=dblarr(t_max)
f_init=dblarr(t_max,n)
f=dblarr(t_max,n)
s=double(0)
for i=0, n-1 do begin
f_init(0,i)=P(i)*B_prob(i,observation,0,B,cell_no)
s=s+f_init(0,i)
;print, 'state is ', i, 'b_prob is ', B_prob(i,observation,0,B,cell_no)
endfor
c[0]=1/s
for i=0, n-1 do begin
f(0,i)=f_init(0,i)*c[0]
endfor

for t=0, (t_max-2) do begin
s=double(0)
for j=0, (n-1) do begin      ; f(t,i),A(i,j)
f_init(t+1,j)=sum(0,n-1,t,j,f,A)*B_prob(j,observation,(t+1),B,cell_no) ;i goes
from 0 to n-1
s=s+f_init(t+1,j)
endfor
c[t+1]=1/s
for j=0, (n-1) do begin
f(t+1,j)=f_init(t+1,j)*c[t+1]
endfor
endfor
```

```
;Let's define the backward variables: BACK
Back_init=dblarr(t_max,n)
Back=dblarr(t_max,n)
for i=0, n-1 do begin
Back_init(t_max -1,i)=1
Back(t_max -1,i)=Back_init(t_max -1,i)*c[t_max-1]
endfor
for t=t_max-2, 0, -1 do begin
for i=0, (n-1) do begin
s=double(0)
for j=0,n-1 do begin
s=s + A(i,j)*B_prob(j,observation,(t+1),B,cell_no)*Back(t+1,j)
endfor
Back_init(t,i)=s
Back(t,i)=Back_init(t,i)*c[t]
endfor
endfor
;print, 'these are backward variables'
;print, Back
;print, 'space'


;Now let's define the sequence of probabilities xsi(t,i,j) - it is the
;probaility of being at state i at time t and at state j at time t+1
xsi=dblarr(t_max,n,n)
for t=0,t_max-2 do begin
s=double(0)
for i=0,n-1 do begin
for j=0,n-1 do begin
s=s+F(t,i)*A(i,j)*B_prob(j,observation,(t+1),B,cell_no)*Back(t+1,j)
endfor
endfor
for i=0,n-1 do begin
for j=0,n-1 do begin
xsi[t,i,j]=(F(t,i)*A(i,j)*B_prob(j,observation,(t+1),b,cell_no)*Back(t+1,j))/s
endfor
endfor
endfor
;print, 'This is xsi'
;print, xsi
;print, 'space'


;Gamma(t,i) is the probability of being at state i at t given the
;observation and the model
Gamma=Dblarr(t_max,n)
for t=0, t_max -1 do begin
for i=0, n-1 do begin
s=double(0)
for j=0, n-1 do begin
s=s+xsi(t,i,j)
endfor
Gamma[t,i]=s
endfor
```

```
endfor
;print, 'This is gamma'
;print, gamma
;print, 'space'

;Now we will begin the reestimation procedures of initial parameters P,A,B

;reestimation of P
for i=0, n-1 do begin
P(i)=gamma(1,i)
endfor

;Reestimation of A
for i=0, n-1 do begin
for j=0, n-1 do begin
s1=double(0)
s2=double(0)
for t=0, t_max-2 do begin
s1=s1+Xsi(t,i,j)
s2=s2+Gamma(t,i)
endfor
A(i,j)=s1/s2
endfor
endfor

;Reestimation of B
;for j=0, n-1 do begin
;for k=0, m-1 do begin
;s1=double(0)
;s2=double(0)
;for t=0, t_max-1 do begin
;if (observation(t) eq k) then begin
;s1=s1+Gamma(t,j)
;endif
;s2=s2+Gamma(t,j)
;endfor
;B(j,k)=s1/s2
;endfor
;endfor

;Reestimation of B
for j=0, n-1 do begin
for k=0, m-1 do begin
s1=double(0)
s2=double(0)
for t=0, t_max-1 do begin

s1=s1 + Gamma(t,j)*(observation[k,t])

s2=s2+Gamma(t,j)
endfor
B(j,k)=s1/s2
endfor
endfor
```

```
end
; to sum n elements, used by hidden_markov_event.pro


function sum, x,y,t,j,f,A

;s will be used to denote the sum

s=f(t,0)*A(0,j)
for i=(x+1), y do begin
s=s + f(t,i)*A(i,j)
endfor

return, s
end
```

APPENDIX C

```
;Name:  medsub.pro
;Description:  This uses the running median filter to subtract the baseline
;etc.
;Variables:  medsize is the size of the window used in calculating the filter
;             cellsin is a cells X frames array that contains the unfiltered
data
;             cellsout has the cells after the base has been subtracted from
them.

pro medsub, cellsin, medsize, cellsout

cellsout=float(cellsin)
ncells=n_elements(cellsin(*,0))
nframes=n_elements(cellsin(0,*))
tmp1=fltarr(nframes+2*medsize)
for i=0, ncells-1 do begin
      tmp1(medsize:medsize+nframes-1)=reform(cellsin(i,*))
      ;tmp1=(tmp1-median(tmp1,medsize))>0
      tmp1=abs(tmp1-median(tmp1,medsize))
      cellsout(i,*)=tmp1(medsize:medsize+nframes-1)
endfor
end
```

```
pro nonnegfac_converge, x, w1,v1

; Iteration for computing nonnegative factorisation
; x = Matrix; w1, v1 = nonnegative factors.
; x = N x M
; w1 = N x R
; v1 = R x M

wv=w1#v1
xwv=x*0
nz=where(x gt 0)
eps=1e-20
xwv(nz)=x(nz)/(wv(nz)+eps)

v1=v1*(transpose(w1)#xwv)

xwv=xwv*0
wv=w1#v1
xwv(nz)=x(nz)/(wv(nz)+eps)

w1=w1*(xwv#transpose(v1))
norm=rebin(transpose(total(w1,1))+eps,n_elements(w1(*,0)),n_elements(w1(0,*)))
w1=w1/norm

end
```

```
pro nonnegsvd_converge, mat, frac, u, v, nkeep, niter, errors
;svd, mat, w, u, v
;index=reverse(sort(w))
;w=w(index)
;u=u(*,index)
```

```
;v=v(*,index)
sz=size(mat)
x=sz(1)
y=sz(2)
R=nkeep
noise=randomu(seed,R,y)
noise2=randomu(seed,x,R)
w=fltarr(x,R)+.01*noise2
for i=0, R-1 do w(i,i)=1
v=mat(0:R-1,*)+.1*noise
;u=u(*,0:R-1)
;v=v(*,0:R-1)
;for t=0, R-1 do begin
      ;u(*,t)=u(*,t)*sqrt(w(t))
      ;v(*,t)=v(*,t)*sqrt(w(t))
;endfor
;w=abs(u)
;v=abs(transpose(v))
errors=fltarr(niter+1)

for j=0,niter do begin
nonnegfac_converge,mat,w,v
recon=w#v
error=fltarr(R)
norm=total(mat^2)
for i=0,R-1 do error(i) = total((mat-w(*,i)#v(i,*))^2)/norm
srt=sort(error)
frac=1-error(srt)
u=w(*,srt)
v=v(srt,*)
errors(j)=total((mat-recon)^2)/norm
endfor
end
```

```
;NAME:scalculate_svd
;DESCRIPTION: using this, the user finds the nonnegsvd by inputing the number of
iterations and
;the number of modes to keep.   (Though it is called calculate_svd, it is a
different linear
;factorization- nonnegative one.., but close to the general SVD).

pro scalculate_svd
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common pp_com, tp_pixel_array

      base=WIDGET_BASE(/column,title='Perform Non-negative SVD')
      draw=WIDGET_DRAW(base, xsize=400, ysize=400)
      modeskeep=CW_FIELD(base, title='Enter the number of modes that should be
kept:', $
            VALUE=cell_no, /integer, /return_events)
      niterations=CW_FIELD(base, title='Number of iterations:', $
            VALUE=100, /integer, /return_events)
      button=WIDGET_BUTTON(base, value='Calculate SVD', uvalue='button_pressed')
      widget_control, /realize, base
      state={modeskeep:modeskeep, niterations:niterations}
```

```
      widget_control, widget_info(base, /child), set_uvalue=state
      xmanager, 'scalculate_svd', base
      end
```

```
;NAME: scalculate_svd_event
;DESCRIPTION: This is the event handler for scalculate_svd..  It calls
nonnegsvd_converge which calls nonnegfac_converge based on the
;number of iterations and the number of modes to keep. **Make sure to do all the
required error checking...**

pro scalculate_svd_event, event
common pp_com, tp_pixel_array
common svds_com, w, u, v, errors, mkeep, imu
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common flags, svd_calculated, cells_defined
common vis_svdcom, visbase, visbase2
common local, imvar                                                    ;unused
commons such as this should be removed ;this makes it readable from 'Choose
Cells' too
common stockdata, symbol_array, date_array

      stateholder=WIDGET_INFO(event.handler, /Child)
      print, visbase, visbase2
      vismanaged=WIDGET_INFO(visbase, /MANAGED)       ;if this is 1, the general
visualization widget is open
      vismanaged2=WIDGET_INFO(visbase2, /MANAGED)     ;if this is 1, the spatial
visualization widget is open
      WIDGET_CONTROL, stateholder, GET_UVALUE=state   ;find what is in /Child
      WIDGET_CONTROL, state.modeskeep, GET_VALUE=mkeep
      WIDGET_CONTROL, state.niterations, GET_VALUE=niter
      Widget_control, event.id, get_uvalue=uval
      ;if ((vismanaged eq 1) and (mkeep ne mkeepold)) OR ((vismanaged2 eq 1) and
(mkeep ne mkeepold2)) then begin          ;number of modes being calculated is
not number being showed in visualization.general
      ;     mess=WIDGET_MESSAGE('A visualization widget is open with a capacity
for a different number of modes than you currently specify!  Change the number
of modes, or close the visualization widget and repeat.', /ERROR)
      if ((vismanaged eq 1) OR (vismanaged2 eq 1)) then begin
            mess=WIDGET_MESSAGE('To avoid conflicts with the number of modes
specified, please shut your visualization widgets.',/INFORMATION)
      endif else begin
      if uval eq 'button_pressed' then begin
         if (mkeep gt cell_no) then begin
             mess=DIALOG_MESSAGE('Number of modes cannot be more than the number
of cells!',/ERROR)
         endif else begin
            nonnegsvd_converge, tp_pixel_array, w, u, v, mkeep, niter, errors
            ;imu=fltarr(x_size,y_size,mkeep)
            ;for i=0,mkeep-1 do begin
                  ;for this to work with stocks->
                  ;im1=fltarr(x_size,y_size)
                  ;imu(*,*,i)=imvar
                  ;im1(location(*).coord(0),location(*).coord(1))=u(*,i)
;reconstruct to dimensions of image
                        ;for j=0, cell_no-1 do begin              ;draw the
boxes
```

```
                        ;     imu(location(j).coord(0)-
box_size:location(j).coord(0)+box_size,location(j).coord(1)-
box_size:location(j).coord(1)+box_size,i)=im1(location(j).coord(0),location(j).c
oord(1))
                        ;endfor
            ;endfor
            plot, w, xtitle='Mode Number', ytitle='Power Contribution'
            svd_calculated=1
          endelse
        endif

endelse
end
```

```
; NAME: sload_and_convert_excelfile
;; This file reads MS Excel file with stock data of the following format
;; date     ticker1     ticker2     ...   ...
;; value    close1      close2      ...   ...
;;    ..    ..          ..          ..    ..
;; This file creates the following arrays:
;; symbol_array (string) - first row of the excle file minus first value
;; date_array (long or string or date) - first column of the excel file minus
the first value
;; pixel_array (float) - all the rest
;; pixel_array later gets transformed by calculating deltaF/F - under the same
name

pro sload_and_convert_excelfile
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
 total_frame_no, time_resolution, x_size, y_size, box_size
;common mother_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name,$
;     total_frame_no, time_resolution, x_size, y_size, box_size
;common test, str_ing, state3
common old_skool_data, original_data
;common flags, cells_defined, spikes_defined, single_plot_defined,
correl_coef_defined
common flags, svd_calculated, cells_defined
common with_choose_cells_com, threshold1, threshold2, rms_threshold
common textfile_vars, text_flag, diode_array, max_num_of_diodes
common stockdata, symbol_array, date_array


      message_dialog=WIDGET_MESSAGE("This procedure loads stock data, it creates
symbol_array, date_array, and pixel_array", /INFORMATION)

      filename=dialog_pickfile(/read, file=('stocks.slk'), get_path=filepath1)
      if (filename eq '') then begin
            message_dialog=WIDGET_MESSAGE("No data read.", /INFORMATION)
      endif else begin
            close, /all
            openr, 1, filename


            ;use excel import function to fill all the required arrays
            symbol_array= read_sylk(filename, /ARRAY,nrows=1, startcol=1)
```

```
            date_array=read_sylk(filename, /ARRAY, ncols=1,
startrow=1,/uselongs)
            data = read_sylk(filename, /ARRAY, startrow=1, startcol=1)
            close, /all

            cell_no = n_elements(symbol_array)
            frame_no = n_elements(date_array)
            ;state3={frame:frame}
            ;WIDGET_CONTROL, state3.name, SET_VALUE=strmid(filename,
strlen(filepath1))
            ;WIDGET_CONTROL, state3.frame, SET_VALUE=frame_no
            ;WIDGET_CONTROL, state3.time, SET_VALUE=time_resolution

            ;; build 'pixel_array'
            pixel_array = fltarr(cell_no, frame_no)
            for j=0, frame_no -1 do begin
                  for i = 0, cell_no - 1 do begin
                        pixel_array[i,j] = data[j,i]
                  endfor
            endfor
            original_data = pixel_array
            ;the following finds delta F over F
            deltaf=fltarr(cell_no, frame_no)

            for i = 0, cell_no - 1 do begin
                  for j=1, frame_no-1 do begin
                        ;deltaf[i,j]=100*(pixel_array[i,j] - pixel_array[i,j-
1])/pixel_array[i,j-1]
                        deltaf[i,j]=pixel_array[i,j]
                  endfor
            endfor
            for i = 0, cell_no - 1 do begin
                  deltaf[i,0] = 0
            endfor

            pixel_array=deltaf


            ;; initialize other variables
            yes_no_values = intarr(cell_no, frame_no)
            coef = fltarr(cell_no, cell_no)
            rms_threshold = fltarr(cell_no)
            threshold1 = fltarr(cell_no)
            threshold2 = fltarr(cell_no)
            for cell = 0, cell_no - 1 do begin
                  rms_threshold[cell] = 2.0
                  threshold1[cell] = 2.0
                  threshold2[cell] = 3.0
            endfor
            cells_defined = 1
            ;spikes_defined = 0
            ;single_plot_defined = 0
            ;correl_coef_defined = 0
            ;box_size = 2
            ;x_size = max(x_pos_data) + 350
            ;y_size = max(y_pos_data) + 350
```

```
            ;; locations are shifted up and to the right by 300 so that all
locations are positive (and thus can be plotted correctly)
            ;location = replicate({struct, coord:intarr(2), size:0,
half_side:0.00}, cell_no)
            ;for i = 0, (cell_no - 1) do begin
            ;     location(i).size = cell_no
            ;     location(i).half_side = 1
            ;     location(i).coord[0] = x_pos_data[diode_array[i] - 1] + 300
            ;     location(i).coord[1] = y_pos_data[diode_array[i] - 1] + 300
            ;endfor

            text_flag = 1

            message_dialog=WIDGET_MESSAGE("Finished reading data.",
/INFORMATION)
      endelse
end
```

```
;Name: spreprocess_wid
;Decription:  This is the widget that allows you to preprocess your data
(smooth, medsub, abs)
;Modification History

pro spreprocess_wid
      common svd_com, pixel_array, yes_no_values, coef, location, cell_no,
frame_no, file_name, direct,$
      total_frame_no, time_resolution, x_size, y_size, box_size
      base3=WIDGET_BASE(/column, title='Preprocess')
      draw5=WIDGET_DRAW(base3,xsize=550, ysize=300)
      slide2=WIDGET_SLIDER(base3,minimum=1,maximum=cell_no, title='Stock
Number')
      button=cw_bgroup(base3, /row, ['Plot','Median Subtract', 'Smooth', 'Take
Absolute Value', 'Return Original Values'],/return_name, UVALUE=0)
      ;stockwin=CW_FIELD(base3, title='Stock                                    ')
      medwin=CW_FIELD(base3, title='Running Window Width for Median
Subtraction',VALUE=20, /INTEGER, /RETURN_EVENTS)
      smoothwin=CW_FIELD(base3, title='Width of Smoothing Window
',VALUE=3, /INTEGER, /RETURN_EVENTS)
      state1={slide:slide2, medwin:medwin, smoothwin:smoothwin, draw:draw5}
      widget_control, base3, /realize
      widget_control, base3, set_uvalue=state1
      xmanager, 'spreprocess_wid', base3
end
```

```
;Name: spreprocess_wid_event
;Description: Widget handler

pro spreprocess_wid_event, event
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common pp_com, tp_pixel_array

      widget_control, event.top, get_uvalue=state3
      WIDGET_CONTROL, state3.slide, GET_VALUE=cell_numb
      WIDGET_CONTROL, state3.smoothwin, GET_VALUE=swin
      WIDGET_CONTROL, state3.medwin, GET_VALUE=mwin
      WIDGET_CONTROL, state3.draw, GET_VALUE=win_num
```

```
      ;WIDGET_CONTROL, state3.stockwin, SET_VALUE=symbol_array(cell_numb-1)
      if (mwin le 1) then begin
            mess=WIDGET_MESSAGE('Median filter window width must be greater than
1!',/ERROR)
      endif else begin
      if ((swin lt 2) OR (swin ge frame_no)) then begin
            print, frame_no
            mess=WIDGET_MESSAGE('Smoothing window width must be >2 and <Total
number of frames!',/ERROR)
      endif else begin
      if ((size(event.value))[1] eq 7) then begin  ;; i.e. if we've pressed a
button which returns a string as its value...
            Case event.value Of

            'Smooth': BEGIN
            wset, win_num
            for i=0, cell_no-1 do begin
                  tp_pixel_array(i,*)=smooth(tp_pixel_array(i,*),swin)
            endfor
            ;plot, findgen(frame_no)*time_resolution, tp_pixel_array(cell_numb-
1,*)
            plot, findgen(frame_no), tp_pixel_array(cell_numb-1,*),
ytitle='Delta F/F', xtitle='Days'
            END

            'Median Subtract':BEGIN
            wset, win_num
            medsub, tp_pixel_array, mwin,tp_pixel_array

            ;plot, findgen(frame_no)*time_resolution, tp_pixel_array(cell_numb-
1,*)
            plot, findgen(frame_no), tp_pixel_array(cell_numb-1,*),
ytitle='Delta F/F', xtitle='Days'
            END

            'Take Absolute Value': BEGIN
            wset, win_num
            tp_pixel_array=abs(tp_pixel_array)
            ;plot, findgen(frame_no)*time_resolution, tp_pixel_array(cell_numb-
1,*)
            plot, findgen(frame_no), tp_pixel_array(cell_numb-1,*),
ytitle='Delta F/F', xtitle='Days'
            END

            'Return Original Values': BEGIN
            wset, win_num
            tp_pixel_array=pixel_array
            ;plot, findgen(frame_no)*time_resolution, tp_pixel_array(cell_numb-
1,*)
            plot, findgen(frame_no), tp_pixel_array(cell_numb-1,*),
ytitle='Delta F/F', xtitle='Days'
            END

            'Plot': BEGIN
            wset, win_num
            ;plot, findgen(frame_no)*time_resolution, tp_pixel_array(cell_numb-
1,*)
```

```
            plot, findgen(frame_no), tp_pixel_array(cell_numb-1,*),
ytitle='Delta F/F', xtitle='Days'
            END

            ELSE:
            ENDCASE
      endif else begin
                  wset, win_num
                  ;plot, findgen(frame_no)*time_resolution,
tp_pixel_array(cell_numb-1,*)
                  plot, findgen(frame_no), tp_pixel_array(cell_numb-1,*),
ytitle='Delta F/F', xtitle='Days'
            endelse
      endelse
      endelse
      end
```

```
; Name: ssuperimposed_plots
; Description: This program plots the deltaF/F values of a single cell per time.

pro ssuperimposed_plots, delta_f_values, cell_number, frame_number

!p.multi=0
window, 2
;plot, findgen(frame_number)*time_resolution, delta_f_values(0, *),
yrange=[min(delta_f_values)-2,max(delta_f_values)+2],$
;           title='All cells', ytitle='ÆF/F', xtitle='time in seconds'
plot, findgen(frame_number), delta_f_values(0, *), yrange=[min(delta_f_values)-
2,max(delta_f_values)+2],$
            title='All stocks', ytitle='ÆF/F', xtitle='Daily market closes'
xyouts, -6, 0, 1
if cell_number gt 1 then begin
for cell_numb=1, cell_number-1 do begin
       ;oplot, findgen(frame_number)*time_resolution,
(delta_f_values(cell_numb,*))
       oplot, findgen(frame_number), (delta_f_values(cell_numb,*))
;xyouts, -6, cell_numb*20, cell_numb+1
a=(delta_f_values(cell_numb,*) eq max((delta_f_values(cell_numb,*))))
a=where(a eq 1)
xyouts, a,max((delta_f_values(cell_numb,*))), cell_numb+1
endfor
endif
end
```

```
;NAME:  SSVD_gui
;DESCRIPTION:  This widget will be used to load data, pre-process,
;perform an SVD, and then visualize the modes created.

pro ssvd_gui
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common flags, svd_calculated, cells_defined
common vis_svdcom, visbase, visbase2

visbase=long(-1)
visbase2=long(-1)
svd_calculated=0                          ;set flags to 0
```

```
cells_defined=0
base=WIDGET_BASE(/column, title='SVD STOCK GUI')
stuff= {cw_pdmenu_s, flags:0, name:''}
details=[{ cw_pdmenu_s, 0, 'Previous Experiment'}, $
       ;{ cw_pdmenu_s, 0, 'Choose Cells' }, $
       { cw_pdmenu_s, 0, 'Load Data'}, $   ;for the new import of stock data
       { cw_pdmenu_s, 0, 'Superimpose Traces'},$
       { cw_pdmenu_s, 0, 'Preprocess'},$
       { cw_pdmenu_s, 0, 'SVD'},$
       { cw_pdmenu_s, 1, 'Visualize'}, $
       { cw_pdmenu_s, 0, 'Across Stocks'}, $
       { cw_pdmenu_s, 2, 'Within Modes'}]
       ;{ cw_pdmenu_s, 0, 'General'},$
       ;{ cw_pdmenu_s, 2, 'With Locations'}]
pull_down=cw_pdmenu(base, details, /return_full_name, UVALUE=12)

stuff2= {cw_pdmenu_s2, flags:0, name:''}
details=[{ cw_pdmenu_s, 0, 'Save Analysis'}, $
       { cw_pdmenu_s, 0, 'Color'}, $
       { cw_pdmenu_s, 0, 'Retall'}, $
       { cw_pdmenu_s, 0, 'Exit'}]
pull_down2=cw_pdmenu(base, details, /return_full_name, UVALUE=13)


;frameid=CW_FIELD(base, title='Total Number of Frames    ',VALUE=total_frame_no,
/INTEGER, $
;     /RETURN_EVENTS)
;timeid=CW_FIELD(base, title='Time Resolution in seconds',VALUE=time_resolution,
/FLOATING, /RETURN_EVENTS)
;don't need time resolution for this..
;state={frame:frameid, time:timeid}
widget_control, /realize, base
;widget_control, widget_info(base, /Child), SET_UVALUE=state
widget_control, widget_info(base, /Child)
xmanager, 'ssvd_gui', base
end
```

```
; Name:  ssvd_gui_event
; Synopsis: ssvd_gui_event, event
; Description: This is the event handler for the ssvd_gui window.

pro ssvd_gui_event, event
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common pp_com, tp_pixel_array
common flags, svd_calculated, cells_defined
common stockdata, symbol_array, date_array

stateholder=WIDGET_INFO(event.handler, /Child)
;WIDGET_CONTROL, stateholder, GET_UVALUE=state  ;find what is in /Child
;WIDGET_CONTROL, state.frame, GET_VALUE=total_frame_no
;WIDGET_CONTROL, state.time, GET_VALUE=time_resolution
Case event.value OF

      'Previous Experiment':BEGIN
       ;tempstate=state
```

```
            data_file=dialog_pickfile(/read, title='Select the saved variables
file', GET_PATH=filepath1)
            if (data_file eq '') then begin
                mess=WIDGET_MESSAGE('No saved variables file specified',
/INFORMATION)
            endif else begin
                restore, data_file
                mess=WIDGET_MESSAGE('Done loading variables', /INFORMATION)
                cells_defined=1
                ;WIDGET_CONTROL, tempstate.frame, SET_VALUE=total_frame_no
                ;WIDGET_CONTROL, tempstate.time, SET_VALUE=time_resolution
;not needed anymore for stocks
            endelse
      END

      'Load Data': BEGIN
            sload_and_convert_excelfile
      END

      'Superimpose Traces': BEGIN
            ;if ((total_frame_no eq 0) or (time_resolution eq 0)) then begin
            ;     mess=WIDGET_MESSAGE('You must enter the total number of frames
and time resolution before proceeding!', /ERROR)
            ;endif else begin
            if (cells_defined eq 1) then begin
                tp_pixel_array=pixel_array
                ;superimposed_plots, tp_pixel_array, cell_no, frame_no,
time_resolution
                ssuperimposed_plots, tp_pixel_array, cell_no, frame_no
            endif else begin
                mess=WIDGET_MESSAGE('Buddy, load data before plotting.',
/ERROR)
            ;endelse
            endelse
      END

      'Preprocess': BEGIN
      ;     if ((total_frame_no eq 0) or (time_resolution eq 0)) then begin
      ;          mess=WIDGET_MESSAGE('You must enter the total number of frames
and time resolution before proceeding!', /ERROR)
        ;  endif else begin
            if (cells_defined eq 1) then begin
                spreprocess_wid
            endif else begin
                mess=WIDGET_MESSAGE('Load data before plotting.', /ERROR)
            ;endelse
            endelse
       END

      'SVD': BEGIN
      ;     if ((total_frame_no eq 0) or (time_resolution eq 0)) then begin
      ;          mess=WIDGET_MESSAGE('You must enter the total number of frames
and time resolution before proceeding!', /ERROR)
      ;     endif else begin
            if (cells_defined eq 1) then begin
                scalculate_svd
            endif else begin
```

```
                mess=WIDGET_MESSAGE('Load data before calculating the SVD.',
/ERROR)
            ;endelse
            endelse
        END

       ;'Visualize.General': BEGIN
        'Visualize.Across Stocks': BEGIN
            ;if ((total_frame_no eq 0) or (time_resolution eq 0)) then begin
            ;     mess=WIDGET_MESSAGE('You must enter the total number of frames
and time resolution before proceeding!', /ERROR)
            ;endif else begin
            if (svd_calculated eq 1) then begin
                  svisualize_wid
            endif else begin
                  mess=WIDGET_MESSAGE('You must calculate the SVD before
visualizing the modes.',/ERROR)
            ;endelse
            endelse
        END

        'Visualize.Within Modes': BEGIN
            if (svd_calculated eq 1) then begin
                  svisualize_wid_locs
            endif else begin
                  mess=WIDGET_MESSAGE('You must calculate the SVD before
visualizing the modes.',/ERROR)
            endelse
         END

       'Color': xloadct

       'Retall':  retall

       'Save Analysis': BEGIN

            data_file=pickfile(/write, file='Variables_svd#_', title='Create the
saved variables file')
            if (data_file eq '') then begin
                        mess=WIDGET_MESSAGE('No saved variables file specified',
/INFORMATION)
                    endif else begin
                        SAVE, /VARIABLES, FILENAME=data_file, all, /verbose
                        mess=WIDGET_MESSAGE('Data from this experiment saved',
/INFORMATION)
                     endelse
          END

      'Exit':BEGIN
            WIDGET_CONTROL, /DESTROY, event.top
        END

       ENDCASE
end
```

```
; Name:  svisualize_wid
```

```
; Description:  This is the widget that shows the spatial and temporal modes of
the data.

pro svisualize_wid
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common svds_com, w, u, v, errors, mkeep, imu
common wid, num_plots
common vis_svdcom, visbase, visbase2
      num_plots=0                    ;no plots have been done, so the first must
be a plot, not Oplot
      visbase=WIDGET_BASE(/column, title='Visualization Widget')
      draw1=WIDGET_DRAW(visbase, xsize=500,ysize=100)
      draw2=WIDGET_DRAW(visbase, xsize=500,ysize=150)
      slide=WIDGET_SLIDER(visbase, value=1, maximum=mkeep, minimum=1,$
            title='Mode Number', uvalue='slider_pressed')
      button1=WIDGET_BUTTON(visbase, value='Plot', uvalue='plot_pressed')
      draw3=WIDGET_DRAW(visbase, xsize=500,ysize=150)
      slide2=WIDGET_SLIDER(visbase, value=1, maximum=cell_no, minimum=1,$
            title='Stock Number', uvalue='slider2_pressed')
      button2=WIDGET_BUTTON(visbase,value='Add Stock to Plot',uvalue='Add')
      button3=WIDGET_BUTTON(visbase,value='Clear Stocks',uvalue='Clear')
      WIDGET_CONTROL, /realize, visbase
      loadct, 27
      state1={draw1:draw1, draw2:draw2, draw3:draw3, slide:slide, slide2:slide2,
button2:button2,$
            button3:button3}
      WIDGET_CONTROL, visbase, set_uvalue=state1
      xmanager, 'svisualize_wid', visbase
end
```

```
; Name: svisualize_wid_event
; Description:  This reads the mode chosen and displays the spatial and temporal
modes for it.

pro svisualize_wid_event, event
common svds_com, w, u, v, errors, mkeep, imu
common wid, num_plots
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common pp_com, tp_pixel_array
common stockdata, symbol_array, date_array

      WIDGET_CONTROL, event.top, get_uvalue=state
      WIDGET_CONTROL, state.draw1, get_value=win1
      WIDGET_CONTROL, state.draw2, get_value=win2
      WIDGET_CONTROL, state.draw3, get_value=win3
      WIDGET_CONTROL, state.slide, get_value=mode_no
      WIDGET_CONTROL, state.slide2,get_value=plotcell
      WIDGET_CONTROL, event.id, get_uvalue=uval
      !y.minor=-1 ;this is so that you can see the values on the y axis-
suppress the tick-marks
      if ((string(uval) eq 'plot_pressed') OR (string(uval) eq
'slider_pressed')) then begin
            wset, win2 & plot, v(mode_no-1,*), xtitle='Days',
ytitle='Amplitude', title='Temporal Mode'
```

```
            wset, win1 & plot, findgen(cell_no)+1, u(*,mode_no-1), psym=2,
xtitle='Stock Number', title='Spatial Mode'
            ;wset, win1 & plot, u(*,mode_no-1),
Xtickname=symbol_array,xticks=cell_no, psym=2, xtitle='Stock', title='Spatial
Mode'
            ;wset, win1 & bar_plot, u(*,mode_no-1), barnames=symbol_array
            num_plots=0
      endif
      if (string(uval) eq 'Add') then begin                              ;Add
Cell to Plot was pressed
            if (num_plots eq 0) then begin
                  wset, win3 & plot, tp_pixel_array(plotcell-1,*),
title='Superimposed Plots'
                  num_plots=1
            endif else begin
                  wset, win3 & oplot, tp_pixel_array(plotcell-1,*),
color=num_plots*100
                  num_plots=num_plots+1
            endelse
      endif

      if (string(uval) eq 'Clear') then begin                    ;Clear Cells
was pressed
                  num_plots=0
                  wset, win3 & erase
      endif
      !y.minor=0
end
```

```
; Name: svisualize_wid_locs
; Description:  This shows the spatial modes of the components with specific
locations

pro svisualize_wid_locs
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common svds_com, w, u, v, errors, mkeep, imu
common vis_svdcom, visbase, visbase2
common locs, n_rows, n_columns
      n_columns=round(sqrt(cell_no))            ;numb e   r of columns to make
closest to sqre
      n_rows=ceil(cell_no*1.0/n_columns)
      visbase2=WIDGET_BASE(/column, title='Scroll through the spatial modes')
      ;draw=WIDGET_DRAW(visbase2, xsize=x_size, ysize=y_size)
      draw=WIDGET_DRAW(visbase2, xsize=n_rows*70, ysize=n_columns*70)
      slide=WIDGET_SLIDER(visbase2, value=1, maximum=mkeep, minimum=1,
title='Mode Number', $
            uvalue='slider_pressed')
      button1=WIDGET_BUTTON(visbase2, value='Plot', uvalue='plot_pressed')
      WIDGET_CONTROL, /realize, visbase2
      state1={draw:draw, slide:slide}
      WIDGET_CONTROL, visbase2, set_uvalue=state1
      loadct, 5
      xmanager, 'svisualize_wid_locs', visbase2
end
```

```
; Name: svisualize_wid_locs_event
; Description:  This will show the spatial locations of the cells from the modes

pro svisualize_wid_locs_event,event
common svd_com, pixel_array, yes_no_values, coef, location, cell_no, frame_no,
file_name, direct,$
total_frame_no, time_resolution, x_size, y_size, box_size
common stockdata, symbol_array, date_array
common svds_com, w, u, v, errors, mkeep, imu
common locs, n_rows, n_columns
cell_numb=0
      WIDGET_CONTROL, event.top, get_uvalue=state
      WIDGET_CONTROL, state.draw, get_value=win
      WIDGET_CONTROL, state.slide, get_value=mode_no
      WIDGET_CONTROL, event.id, get_uvalue=uval
      if ((string(uval) eq 'plot_pressed') OR (string(uval) eq
'slider_pressed')) then begin
            ;loadct, 5
            wset, win

            ;the best way to represent the data seems to be in a matrix built of
symbols
            ;with round(sqrt(totalstocks)) as the number of columns
            ;tvscl, imu(*,*,mode_no-1)
            imu=fltarr(200*n_columns, 250*n_rows)
            index1=(10+n_columns-1)*n_columns
            ;for i=20, 5*index1, 5*n_columns  do begin       ;to give it space
to draw a box
            ;     for j=20, (10+n_rows-1)*5*n_rows, 5*n_rows do begin
            ;for i=10, (10+n_columns-1)*10, 10 do begin
            for i=50, (1+n_rows-1)*50, 50 do begin
                  for j=60, (1+n_columns-1)*60, 60 do begin
                        cell_numb=cell_numb+1
                        if (cell_numb le cell_no) then begin
                        imu(i-5:i+5, j-5:j+5)=u(cell_numb-1,mode_no-1)  ;specify
box size later
                        endif
                        ;xyouts, i-50, j-25, cell_numb, /device
                  endfor
            endfor

            ;for i=0, cell_no-1 do begin
            ;     xyouts, location(i).coord(0), location(i).coord(1),
string(i+1), color=100,/device
            ;endfor
            tvscl, imu        ;now write stock symbols on top
            cell_numb=0
                  for i=50, (10+n_rows-1)*50, 50 do begin
                              for j=60, (1+n_columns-1)*60, 60 do begin
                              cell_numb=cell_numb+1
                              ;print, cell_numb
                              if (cell_numb le cell_no) then begin
                              ;xyouts, i-50, j-25, cell_numb, /device
                              xyouts, i-35, j+15, cell_numb, /device,
charsize=.8
                              xyouts, i-15, j-25, symbol_array(cell_numb-1),
/device, charsize=1.2 ; , color=u(cell_numb-1,mode_no-1)*5
```

```
                                endif
                                endfor
                        endfor
        endif

end
```